

AFRL-MN-EG-TR-2000-7045

DIGITAL HOLOGRAPHY

Rodney M. Powell

Air Force Research Laboratory Munitions Directorate
Lethality & Vulnerability Branch
101 W. Eglin Blvd, Suite 309
Eglin Air Force Base, FL 32542-6810



FEBRUARY 2000

FINAL REPORT FOR PERIOD OCTOBER 1995 - DECEMBER 1998

DISTRIBUTION approved for public release; distribution is unlimited.

AIR FORCE RESEARCH LABORATORY, MUNITIONS DIRECTORATE

Air Force Materiel Command ■ United States Air Force ■ Eglin Air Force Base

THIS DOCUMENT IS UNCLASSIFIED

20001017 027

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

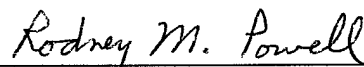
This technical report is releasable to the National Technical Information Services (NTIS). At NTIS it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER



FREDERICK A. DAVIS
Technical Director, Assessment and
Demonstrations Division



RODNEY M. POWELL
Program Manager

If your address has changed, if you wish to be removed from our mailing list, or if your organization no longer employs the addressee, please notify AFRL/MNAL, Eglin AFB FL 32542-6810, to help us maintain a current mailing list.

Do not return copies of this report unless contractual obligations or notice on a specific document requires that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE FEBRUARY 2000	3. REPORT TYPE AND DATES COVERED FINAL -- OCTOBER 1995 TO DECEMBER 1998	
4. TITLE AND SUBTITLE DIGITAL HOLOGRAPHY			5. FUNDING NUMBERS Contract #: N/A PE: 62602F PR: 2502 TA: 28 WU: 33	
6. AUTHOR(S) Powell, R.M.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lethality and Vulnerability Branch Wright Laboratory Armament Directorate AFRL/MNAL 101 W. Eglin Blvd, Suite 309 Eglin AFB, FL 32542-6810			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) (Program Mgr Name & Ph #) Air Force Research Laboratory, Munitions Directorate Lethality & Vulnerability Branch (AFRL/MNAL) Project Engineer: Mr Rodney M. Powell, (850) 882-2547 101 W. Eglin Blvd., Suite 309 Eglin AFB, FL 32542-6810			10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-MN-EG-TR-2000-7045	
11. SUPPLEMENTARY NOTES NONE				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Distribution approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE (Leave Blank)	
13. ABSTRACT: Holography is frequently used to provide scientific data such as the size and shape of objects. Extracting the desired data from holograms presents a challenge. This is typically done by taking snapshots of the hologram at various viewing angles and using stereoscopic, tomographic back-projection, or other similar type algorithms to reconstruct the three-dimensional image of the object in a computer. This research explores another approach. It is to digitize the hologram directly into a computer. Then, the computer is used to perform a mathematical reconstruction of the three-dimensional image of the object based on the interference fringe pattern. Among the theoretical issues addressed are spatial resolution and depth of field of the reconstructions. Also covered is software program code on how to implement the mathematical reconstruction on a computer. Experimentation is conducted to include creating, digitizing, and mathematically reconstructing holograms to verify the theory and programming.				
14. SUBJECT TERMS holography, digital holography, holographic reconstruction, stereoscopy, tomography, fringe pattern, three-dimensional imaging			15. NUMBER OF PAGES 120	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR	

TABLE OF CONTENTS

	<u>page</u>
1 INTRODUCTION.....	1
2 THEORETICAL BACKGROUND	3
2.1 Mathematics of Holography.....	3
2.1.1 The Object Beam.....	3
2.1.2 Recording a Complex Amplitude Hologram	6
2.1.3 Image Reconstruction.....	7
2.1.4 Separation of Components in Reconstruction.....	10
2.2 Spatial Frequency Due to Interference	11
3 DIGITAL RECONSTRUCTION	14
3.1 Digitization of the Hologram	14
3.1.1 Microscope Objectives and Numerical Aperture	14
3.1.2 Finite Pixel Size	16
3.2 Effect of Limited Scan Area.....	19
3.2.1 Maximum Theoretical Resolution.....	20
3.2.2 Mosaicking.....	21
3.2.3 Depth of Field.....	34
4 EXPERIMENTAL DESIGN.....	38
4.1 Objective of Experiments.....	38
4.2 Components.....	38
4.3 Resolution Test.....	39
4.4 Depth of Field Test.....	41
5 DIGITAL HOLOGRAPHY RECONSTRUCTION PROGRAM.....	46
6 EXPERIMENTAL RESULTS	51
6.1 Microscope Magnification	51
6.2 Unfeasibility of Testing Mosaicking	52
6.3 Film Exposure and Developing.....	53
6.4 Resolution Determination	54
6.5 Depth of Field Demonstration.....	58
6.6 Distortion and Noise Considerations.....	61
7 CONCLUSIONS.....	63

7.1 Project Results.....	63
7.2 Suggestions for Future Research.....	64

APPENDICES

A MOSAICKING ERROR ANALYSIS PROGRAM.....	66
B MOSAICKING PROGRAM AND SUBPROGRAMS.....	69
C DIGITAL RECONSTRUCTION PROGRAM AND SUBPROGRAMS	88
D RELATIVE ERROR PROPAGATION	111
LIST OF REFERENCES	113

1.0 INTRODUCTION

Holography is well known for its fascinating three-dimensional images. In the scientific community, however, holograms are frequently used for obtaining three-dimensional data such as size and shape about the objects of which a hologram is made. A common technique for extracting such data is illustrated in Fig. 1.1. This approach involves taking pictures of the optical image from different viewing angles and storing them in a computer. Then, the computer uses the stored pictures to reconstruct the three-dimensional image of the object.

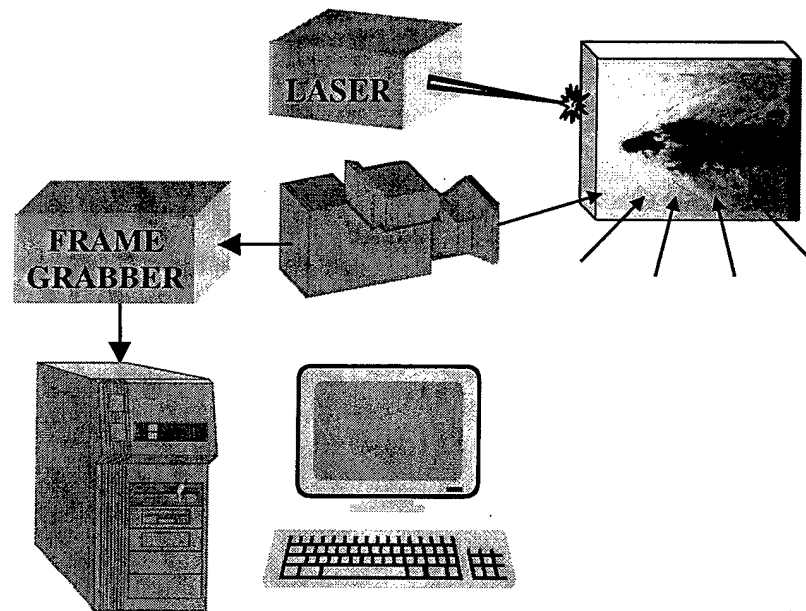


Figure 1.1 Optical reconstruction of holograms.

Reconstruction via digital holography takes a different approach. The steps involved are illustrated in Fig 1.2. The first step is to magnify the hologram. Next, an electronic camera and a frame grabber are used to electronically capture and digitize the hologram for storage in a computer. Finally, the computer is used to mathematically reconstruct the image of the object from the fringe pattern in the hologram.

The impetus for exploring digital holography came as a natural progression from the conventional technique depicted in Fig 1.1. Since a computer had to be used anyway, it was thought that if the hologram itself could be directly digitized into a computer, then the step of using a camera to take pictures from various angles with its associated error of mechanical placement could be eliminated. Additionally, since the hologram itself would be in the computer, this would allow processing of the hologram before the images were formed. The recreation would be a matter of mathematics rather than optics.

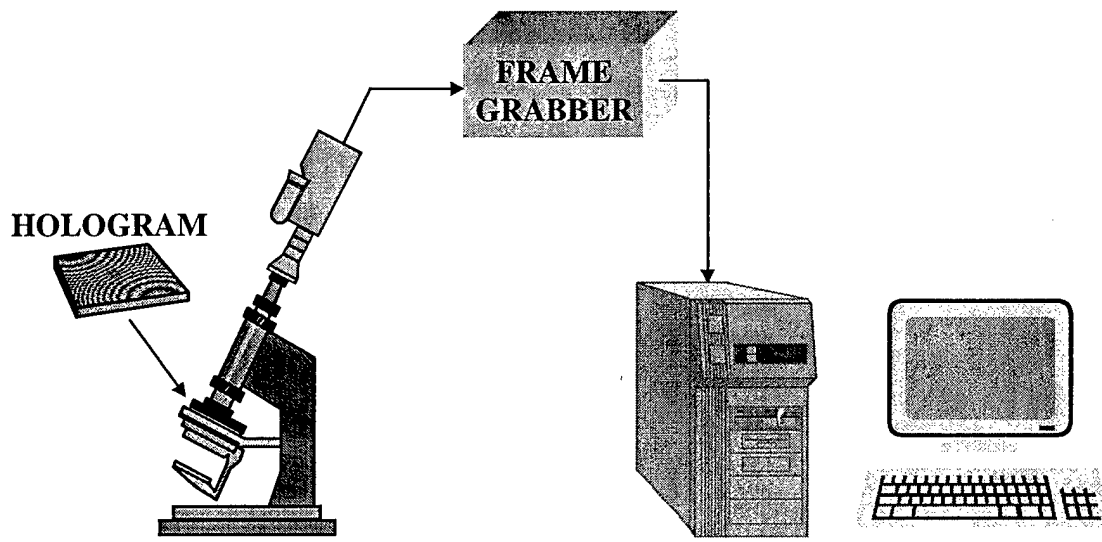


Figure 1.2 Digital reconstruction process.

2.0 THEORETICAL BACKGROUND

Diffraction is the key theory for describing how electromagnetic waves or light propagate through space. Sommerfeld described diffraction as “any deviation of light rays from rectilinear paths which cannot be interpreted as reflection or refraction” [Goodman, 1996, p. 33]. By utilizing the principles of diffraction, one can characterize a wave at locations along its path of propagation. In order to take a complete snapshot of a wave at any particular location, it is necessary to store both the amplitude and phase of the wave. A recording of this type is known as a hologram which means a “total recording” [Goodman, 1996, p. 295].

2.1 Mathematics of Holography

The purpose for presenting the mathematical theory of holography is to allow the quantifying of the holographic process so that its limitations can be predicted. Development begins with an object beam and illustrates how approximations to the diffraction theory can be applied to simplify description of the object beam as it propagates through free space. Eventually, the approximations will enable the description of the object beam, after propagating through some distance d , in terms of the Fourier transform. Next, the method of recording the object beam, both amplitude and phase, will be discussed. Finally, the process of optical reconstruction of the object beam will be explained thus resulting in the recovery of the original object waveform. By laying this theoretical background at this stage, we can then use this background to predict the benefits and limitations of reconstructing the beam digitally.

2.1.1 The Object Beam

Consider Fig. 2.1 which contains an infinite opaque screen with an aperture. It is through this aperture that the object wavefront or beam originates. Additionally, a region of observation is shown. The region of observation is assumed to be a plane parallel to the original screen. Beginning with the Rayleigh-Sommerfeld diffraction formula, the object beam, in the observation plane, can be described by the Huygens-Fresnel relationship [Goodman, 1996, p. 46-66]

$$\mathbf{U}(\xi, \eta) = \iint_{\Sigma} \mathbf{g}(\xi, \eta; x, y) \mathbf{O}(x, y) dx dy, \quad (2.1)$$

where $\mathbf{O}(x, y)$ is the object beam at the screen and $\mathbf{g}(\xi, \eta; x, y)$ is a transformation kernel [Yaroslavskii, 1980, p. 4]. In Eq. (2.1), the transformation kernel is given by

$$\mathbf{g}(\xi, \eta; x, y) = \frac{1}{j\lambda} \frac{e^{jkr_{pp'}}}{r_{pp'}} \cos(\theta). \quad (2.2)$$

$\mathbf{O}(x, y)$ is zero outside the aperture Σ . In Fig. 2.1, P' represents a point in the observation plane for which the wavefront is being calculated and P represents one point in the aperture over which the integration is being carried out. The angle θ is that between the line connecting the points P and P' and the normal of the aperture plane.

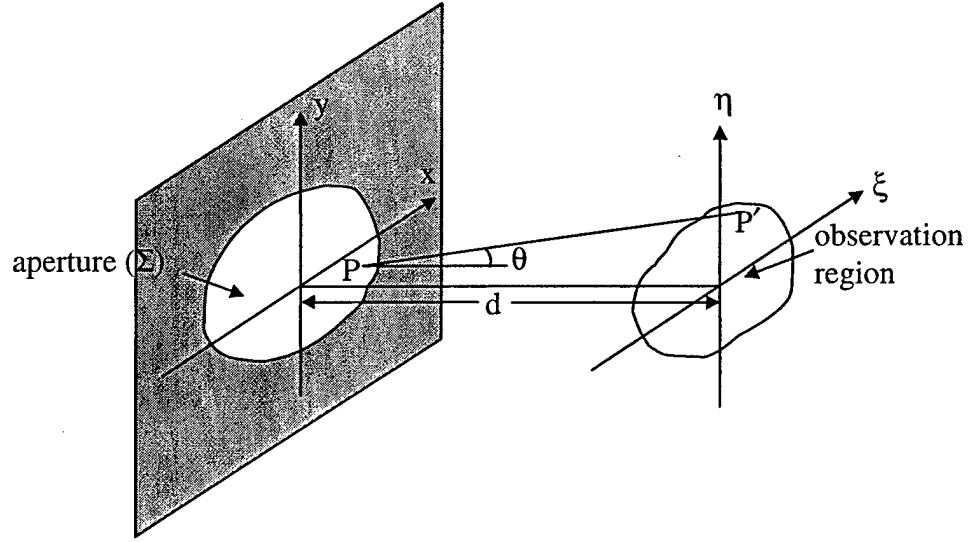


Figure 2.1 Diffraction geometry.

Letting $r_{pp'}$ represent the line between P and P' and d the distance between the planes, then from the geometry,

$$\cos(\theta) = \frac{d}{r_{pp'}}. \quad (2.3)$$

Substituting Eq. (2.3) into Eq. (2.2) yields

$$\mathbf{g}(\xi, \eta; x, y) = \frac{d}{j\lambda} \frac{e^{jkr_{pp'}}}{(r_{pp'})^2}. \quad (2.4)$$

Approximations can be made to reduce the formula into a more usable form [Goodman, 1996, p. 66-67]. If θ is small, then $(r_{pp'})^2$ in the denominator can be approximated as d^2 . However, this approximation cannot be made in the exponential term of the numerator since small changes in $r_{pp'}$ will cause large variations in $e^{jkr_{pp'}}$. Approximations can be made in that term through use of the binomial expansion of a square root as follows. The distance between P and P' is given exactly by

$$r_{pp'} = \sqrt{d^2 + (\xi - x)^2 + (\eta - y)^2} = d \sqrt{1 + \left(\frac{(\xi - x)}{d}\right)^2 + \left(\frac{(\eta - y)}{d}\right)^2}. \quad (2.5)$$

The binomial expansion of a square root is

$$\sqrt{1+b} = 1 + \frac{1}{2}b - \frac{1}{8}b^2 + \dots \quad (2.6)$$

for $|b| < 1$. Assuming the 1st two terms will give an adequate approximation, we have

$$r_{pp'} \cong d \left[1 + \frac{1}{2} \left(\frac{\xi - x}{d} \right)^2 + \frac{1}{2} \left(\frac{\eta - y}{d} \right)^2 \right]. \quad (2.7)$$

Substituting this approximation into Eq. (2.4) and rearranging terms produces

$$\mathbf{g}(\xi, \eta; x, y) \equiv \left(\frac{e^{jkd}}{j\lambda d} \right) e^{\frac{jk}{2d} [(\xi - x)^2 + (\eta - y)^2]} \quad (2.8)$$

which leads to

$$\begin{aligned} \mathbf{U}(\xi, \eta) &= \frac{e^{jkd}}{j\lambda d} \iint_{\Sigma} \mathbf{O}(x, y) \exp \left[\frac{jk}{2d} ((\xi - x)^2 + (\eta - y)^2) \right] dx dy \\ &= \frac{e^{jkd}}{j\lambda d} \exp \left[\frac{jk}{2d} (\xi^2 + \eta^2) \right] \iint_{\Sigma} \mathbf{O}(x, y) \exp \left[\frac{jk}{2d} (x^2 + y^2) \right] \exp \left[-\frac{j2\pi}{\lambda d} (x\xi + y\eta) \right] dx dy. \end{aligned} \quad (2.9)$$

Equation (2.9) is the Fresnel approximation relating the complex optical amplitude distribution in the observation plane to that at the screen [Goodman, 1996, p. 67]. When d is large enough for the approximation, the observer is in the Fresnel or near-field diffraction region. In the last exponential term, the wave number k was substituted with its equivalent $2\pi / \lambda$. Notice that the integral in Eq. (2.9) is just the two-

dimensional forward Fourier transform of $\mathbf{O}(x, y) \exp \left[\frac{jk(x^2 + y^2)}{2d} \right]$, except for the

factor $\frac{1}{\lambda d}$ in the last exponential term. We will explore this factor later when investigating digital reconstruction.

The significance of this development can begin to be appreciated. Computing a Fourier transform digitally on a computer can easily be done through use of the fast

Fourier transform (FFT). Also, the term $\exp \left[\frac{jk(x^2 + y^2)}{2d} \right]$ in Eq. (2.9) can be created

mathematically as can all of the terms in front of the integrals. So, if the wavefront $\mathbf{O}(x, y)$ in Eq. (2.9) is known, then $\mathbf{U}(\xi, \eta)$ can be computed mathematically on a computer.

A further simplification to Eq. (2.9) can be developed and is known as the Fraunhofer or far-field diffraction approximation. If d is very large such that

$d \gg \frac{k(x^2 + y^2)_{\max}}{2}$, then

$$\exp \left[\frac{jk(x^2 + y^2)}{2d} \right] \cong 1 \quad (2.10)$$

which leads to

$$\mathbf{U}(\xi, \eta) = \frac{e^{jkd}}{j\lambda d} \exp\left[\frac{jk}{2d}(\xi^2 + \eta^2)\right] \iint_{\Sigma} \mathbf{O}(x, y) \exp\left[-\frac{j2\pi}{\lambda d}(x\xi + y\eta)\right] dx dy \quad (2.11)$$

Thus, in the Fraunhofer region, the optical complex amplitude is related to $\mathbf{O}(x, y)$ by the forward Fourier transform directly (except for the factor $\frac{1}{\lambda d}$ in the last exponential term as mentioned earlier).

2.1.2 Recording a Complex Amplitude Hologram

To record a hologram, it is necessary to store information about both the amplitude and phase of the object wavefront. The standard technique to accomplish this is by recording the constructive and destructive interference between two coherent wavefronts; this process is known as interferometry. The mathematical description of this technique can be illustrated by computing the superposition of two wavefronts. The wavefront from the object beam, at the film plane, can be described as

$$\mathbf{U}(\xi, \eta) = U(\xi, \eta)e^{-j\phi(\xi, \eta)}, \quad (2.12)$$

where $U(\xi, \eta)$ and $\phi(\xi, \eta)$ are the amplitude and phase of the object beam, respectively. Similarly, the reference beam is given by

$$\mathbf{A}(\xi, \eta) = Ae^{-j\psi(\xi, \eta)}, \quad (2.13)$$

where A and $\psi(\xi, \eta)$ are the amplitude and phase of the reference beam, respectively. By design, the reference beam amplitude is made approximately a constant.

Unfortunately, photographic film cannot record complex values. Film actually records intensity, which is calculated by multiplying one term by its conjugate. For the case where there are two wavefronts incident on the film simultaneously, the film will record the intensity of the sum [Kozma, 1966]. Thus, the intensity of the object beam and reference beam together is

$$\begin{aligned} I(\xi, \eta) &= (\mathbf{U} + \mathbf{A})(\mathbf{U} + \mathbf{A})^* \\ &= |\mathbf{U}|^2 + |\mathbf{A}|^2 + \mathbf{A}^* \mathbf{U} + \mathbf{A} \mathbf{U}^* \\ &= A^2 + U^2(\xi, \eta) + 2AU(\xi, \eta) \cos[\psi(\xi, \eta) - \phi(\xi, \eta)] \end{aligned} \quad (2.14)$$

The last term of Eq. (2.14) shows how both amplitude and phase of the object are preserved. In contrast, if the wavefronts incident upon the film are not coherent, then constructive and destructive interference will not occur. In this case, the film will record purely the addition of the individual intensities as in ordinary photographs [Kreis, 1996, p. 22].

Film does not necessarily record intensity linearly. However, film does typically have a linear portion in its t-E curve. If the exposure time with respect to intensity is chosen so that exposure is in the middle of this linear portion and if A is large enough compared to $U(\xi, \eta)$, then the recording will be confined to the linear portion of the t-E

curve [Kozma, 1966]. Assuming this linearity in the mapping of intensity exposure to amplitude transmittance and that $|A|^2$ is uniform across the recording film surface, then the transmittance is

$$T(\xi, \eta) = t_b + \beta'(|U|^2 + A^*U + AU^*), \quad (2.15)$$

where t_b is a uniform bias resulting from the reference, and β' is the product of the slope β of the film's t-E curve at the bias point and the exposure time [Goodman, 1996, p. 303]. β' is negative for a negative transparency and positive for a positive transparency.

2.1.3 Image Reconstruction

The development in this section will follow closely with [Goodman, 1996, p. 302-309]. Illuminating the hologram by a coherent reconstruction wave $B(\xi, \eta)$ produces light transmitted by the transparency given by:

$$B(\xi, \eta)T(\xi, \eta) = t_b B + \beta' U U^* B + \beta' A^* B U + \beta' A B U^* = S_1 + S_2 + S_3 + S_4. \quad (2.16)$$

If $B = A$, then $S_3 = \beta' |A|^2 U$ which is a constant times a duplicate of the original object.

If $B = A^*$, then $S_4 = \beta' |A|^2 U^*$ which is a constant times the conjugate of the original object. Optically, if the reference wave used in the reconstruction process is the same as the original reference wavefront, then a duplicate of the original diverging object wavefront will be generated. This will produce a virtual image that can be imaged onto a camera through use of a lens. On the other hand, if the conjugate of the original reference wavefront is used, then a real image is created corresponding to an actual focusing of light and so use of a lens is not necessary [Goodman, 1996, p. 301].

Dennis Gabor is accredited with the first appearance of holography in the 1940s. However, it was work with the off-axis reference wave by Leith and Upatnieks in the early 1960s and the invention of the laser that really led to widespread research in the area of holography [Reynolds, 1989, p. 293]. Holograms using an off-axis reference wave are generally referred to as Leith-Upatnieks or offset-reference holograms. It is illustrated in Fig. 2.2. It is made by introducing a separate distinct reference wave at an offset angle to the object-film axis. For this case, the reference wave is described by

$$A(\xi, \eta) = A e^{-j2\pi\alpha\eta}, \quad (2.17)$$

where the spatial frequency α of the reference wave along the film plane is given by

$$\alpha = \frac{\sin \theta}{\lambda}. \quad (2.18)$$

Thus, the intensity distribution across the film for the Leith-Upatnieks hologram is

$$\begin{aligned} I(\xi, \eta) &= (A e^{-j2\pi\alpha\eta} + U(\xi, \eta) e^{-j\phi(\xi, \eta)}) (A e^{-j2\pi\alpha\eta} + U(\xi, \eta) e^{-j\phi(\xi, \eta)})^* \\ &= A^2 + U^2(\xi, \eta) + 2AU(\xi, \eta) \cos(2\pi\alpha\eta - \phi(\xi, \eta)) \end{aligned} \quad (2.19)$$

Figure 2.3 is an actual scanned hologram. It serves to illustrate what is actually recorded on by a hologram. The line structure that can be seen is usually referred to as

the fringe pattern. Note that the picture is an enlargement. Ordinarily, the fringe pattern on a hologram cannot be seen with the eye unless viewed through a microscope.

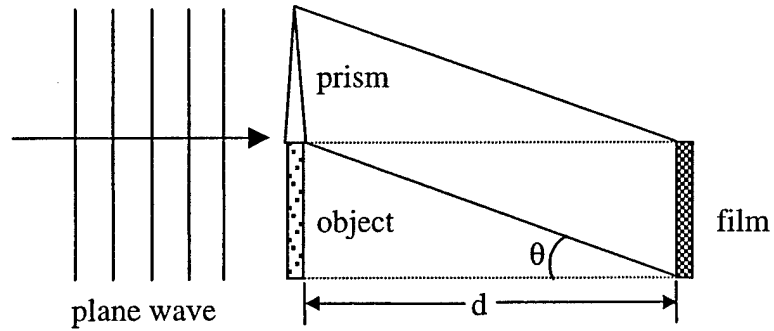


Figure 2.2 Leith-Upatnieks hologram setup.

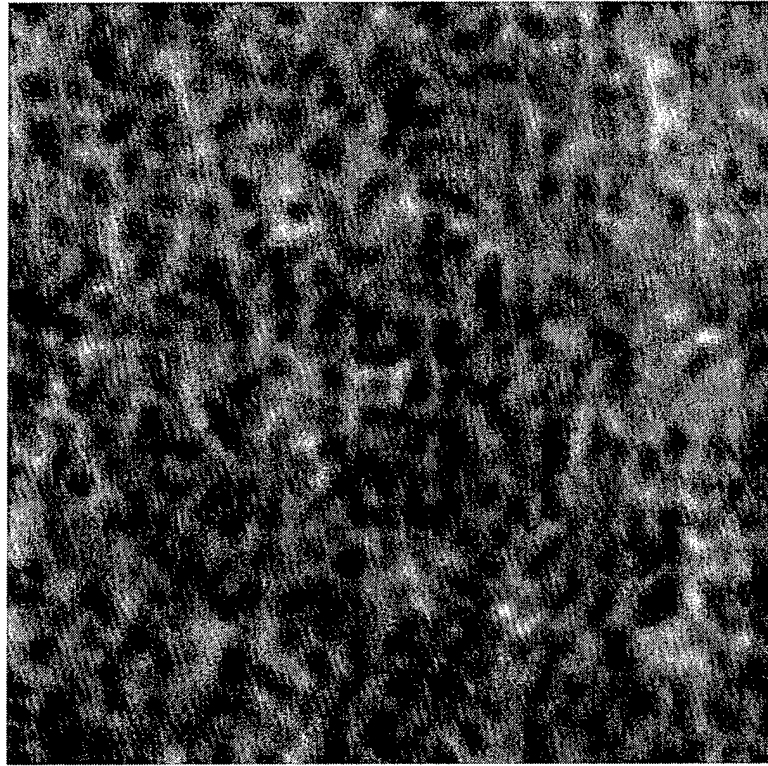


Figure 2.3 Picture of an actual hologram as seen through a microscope.

After the film is developed, its resulting transmittance is

$$T(\xi, \eta) = t_b + \beta'(|\mathbf{U}|^2 + A\mathbf{U}e^{j2\pi\alpha\eta} + A\mathbf{U}^*e^{-j2\pi\alpha\eta}). \quad (2.20)$$

This can be reduced to the following four terms:

$$\begin{aligned}
\mathbf{t}_1 &= t_b \\
\mathbf{t}_2 &= \beta' |\mathbf{U}(\xi, \eta)|^2 \\
\mathbf{t}_3 &= \beta' A \mathbf{U}(\xi, \eta) e^{j2\pi\alpha\eta} \\
\mathbf{t}_4 &= \beta' A \mathbf{U}^*(\xi, \eta) e^{-j2\pi\alpha\eta}
\end{aligned} \tag{2.21}$$

Figure 2.4 illustrates the optical reconstruction of such a hologram. Illumination by a normally incident, uniform plane wave of amplitude B , which is a duplicate of the original reference beam, gives

$$\begin{aligned}
\mathbf{S}_1 &= \mathbf{t}_b B \\
\mathbf{S}_2 &= \beta' B |\mathbf{U}(\xi, \eta)|^2 \\
\mathbf{S}_3 &= \beta' B A \mathbf{U}(\xi, \eta) e^{j2\pi\alpha\eta} \\
\mathbf{S}_4 &= \beta' B A \mathbf{U}^*(\xi, \eta) e^{-j2\pi\alpha\eta}
\end{aligned} \tag{2.22}$$

\mathbf{S}_1 is an attenuated version of the incident reconstruction field and so represents a plane wave traveling down the transparency axis. \mathbf{S}_2 is spatially varying and so contains plane wave components traveling at various angles. \mathbf{S}_3 is proportional to the original object wavefront \mathbf{U} which implies the term generates a virtual image at distance d and angle θ from the transparency. \mathbf{S}_4 is similar to \mathbf{S}_3 . It is proportional to the conjugate of the

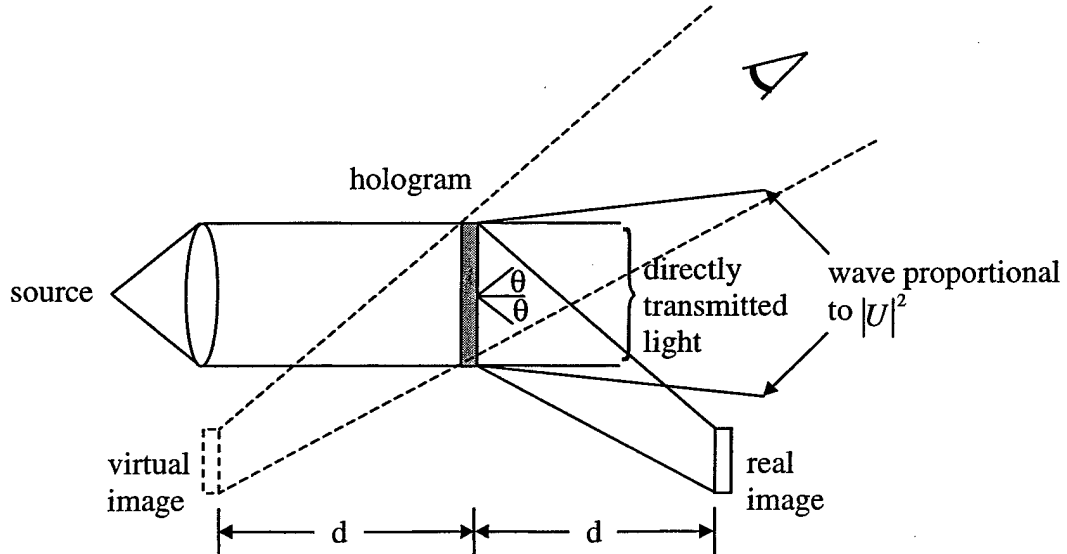


Figure 2.4 Images reconstructed from a Leith-Upatnieks hologram [from Goodman, 1996, p. 307].

object wavefront U^* and produces a real image on the opposite side of the virtual image at an angle $-\theta$ to the transparency axis. These results indicate that both the real and virtual images are angularly separated from each other and the waves S_3 and S_4 . This is due to the angular offset of the reference wave. Also indicated is that an exact duplicate or conjugate of the original reference wave is not always needed for reconstruction.

2.1.4 Separation of Components in Reconstruction

In order to obtain separation of the twin images and the light transmitted along the transparency axis, the spatial-frequency content of S_1 through S_4 above should not overlap. This is achieved through appropriate selection of the reference-offset angle θ . Determination of the minimum angle can be found from the Fourier transform of S_1 through S_4 , which will be denoted as F_1 through F_4 . The transform of the object wavefront will be denoted as F_u . We will assume the highest frequency content of the object is W . From these spectra, the spatial carrier frequency α needed to achieve image separation is found.

Figure 2.5 illustrates these various spectra. The spectrum F_1 is an impulse at origin and thus has zero bandwidth. The spectrum F_2 is proportional to the autocorrelation of F_u and so its bandwidth is $2W$. The spectrum F_3 is proportional to F_u but displaced to center frequency α and so its bandwidth is W . The spectrum F_4 is proportional to a reflected version of F_u and is centered at $-\alpha$. Its bandwidth is also W . From the graphs, $|F_3|$ and $|F_4|$ can be isolated from $|F_2|$ if $\alpha \geq 3W$, or $\sin \theta \geq 3W\lambda$. Thus,

$$\theta_{\min} = \sin^{-1} 3W\lambda. \quad (2.23)$$

If the reference wave is much stronger than the object, then $U \ll A$ and so $|F_2|$ is much smaller than $|F_1|$, $|F_3|$, and $|F_4|$. In this case,

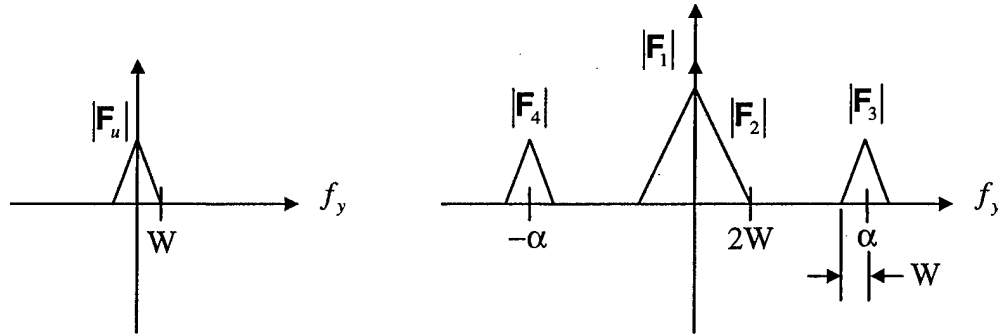


Figure 2.5 Spectra of components from a reconstructed Leith-Upatnieks hologram.

$$\theta_{\min} = \sin^{-1} W\lambda. \quad (2.24)$$

Note that Fig. 2.5 indicates frequency displacement in the y-dimension only since the reference wave was offset only with respect to the y-axis. Similar analysis would apply for an offset angle with respect to the x-axis.

2.2 Spatial Frequency Due to Interference

The issue addressed here is to develop a formula for the spatial frequency that is recorded on the photographic film used to record the hologram. This is necessary for determining the highest spatial frequency recorded on the hologram, which will then dictate the Nyquist sampling frequency required to avoid aliasing. The spatial frequency results from the coherent interference between the reference wave and the object wave. Figure 2.6 illustrates the geometry useful for analyzing this phenomenon. As will become apparent later in the development, it is the maximum angle between the object and the reference wave that determines the highest spatial frequency that will be recorded for any particular setup. The angles indicated in Fig. 2.6 represent the maximum for the geometry in the figure.

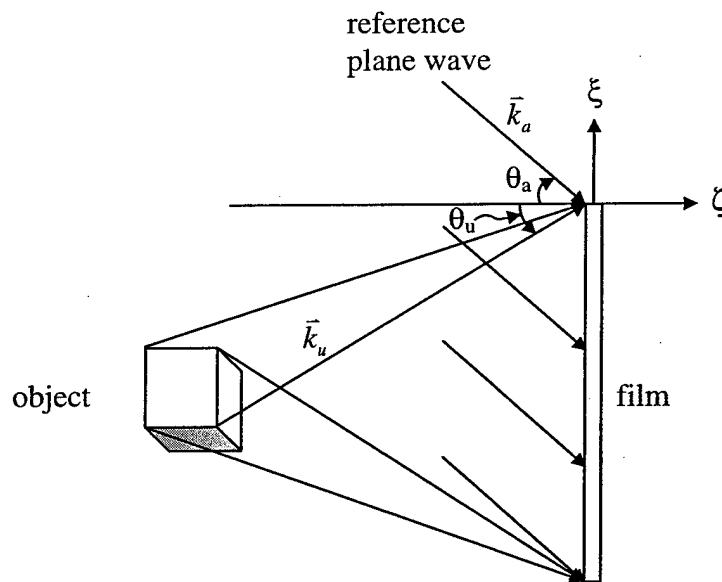


Figure 2.6 Object and reference wave angles.

As already discussed, the film will record the intensity of the superposition of the object and reference waves at each point on the film. The intensity is given by multiplying the sum of the two waves by the conjugate of their sum. For this development, the most general form of representing waves is used to show the full mathematical development.

Defining the intensity as $I(\xi, \eta)$, the object wave at the hologram plane as $\mathbf{U}(\xi, \eta)$, and the reference wave as $\mathbf{A}(\xi, \eta)$, then $I(\xi, \eta)$ is given by the following:

$$\begin{aligned} I(\xi, \eta) &= [\mathbf{A}(\xi, \eta) + \mathbf{U}(\xi, \eta)] \cdot [\mathbf{A}(\xi, \eta) + \mathbf{U}(\xi, \eta)]^* \\ &= \left[A \exp - j(\vec{k}_a \cdot \vec{r} - wt + \phi_a) + U \exp - j(\vec{k}_u \cdot \vec{r} - wt + \phi_u) \right] \\ &\quad \cdot \left[A \exp - j(\vec{k}_a \cdot \vec{r} - wt + \phi_a) + U \exp - j(\vec{k}_u \cdot \vec{r} - wt + \phi_u) \right]^* \end{aligned} \quad (2.25)$$

This simplifies to

$$\begin{aligned} I(\xi, \eta) &= 2A \cos(\vec{k}_a \cdot \vec{r} - wt + \phi_a) \cdot U \cos(\vec{k}_u \cdot \vec{r} - wt + \phi_u) + A^2 \\ &\quad + 2A \sin(\vec{k}_a \cdot \vec{r} - wt + \phi_a) \cdot U \sin(\vec{k}_u \cdot \vec{r} - wt + \phi_u) + U^2 \\ &= 2AU \cos[(\vec{k}_a \cdot \vec{r} - wt + \phi_a) - (\vec{k}_u \cdot \vec{r} - wt + \phi_u)] + A^2 + U^2. \end{aligned} \quad (2.26)$$

From Fig. 2.6,

$$\vec{k}_a = \hat{\xi}k \sin(\theta_a) + \hat{\zeta}k \cos(\theta_a) \quad (2.27)$$

and

$$\vec{k}_u = -\hat{\xi}k \sin(\theta_u) + \hat{\zeta}k \cos(\theta_u). \quad (2.28)$$

Since the position vector is given by

$$\vec{r} = \hat{\xi}\xi + \hat{\eta}\eta + \hat{\zeta}\zeta, \quad (2.29)$$

then

$$\vec{k}_a \cdot \vec{r} = \xi k \sin(\theta_a) + \zeta k \cos(\theta_a) \quad (2.30)$$

and

$$\vec{k}_u \cdot \vec{r} = -\xi k \sin(\theta_u) + \zeta k \cos(\theta_u). \quad (2.31)$$

Substituting Eq. (2.30) and Eq. (2.31) into Eq. (2.26) and using $\zeta = 0$ for the hologram plane leads to

$$\begin{aligned} I(\xi, \eta) &= 2AU \cos\{[k\xi \sin(\theta_a) - wt + \phi_a] - [-k\xi \sin(\theta_u) - wt + \phi_u]\} + A^2 + U^2 \\ &= 2AU \cos\{k\xi[\sin(\theta_a) + \sin(\theta_u)] + \phi_a - \phi_u\} + A^2 + U^2. \end{aligned} \quad (2.32)$$

Since ϕ_a and ϕ_u are constant phase terms, then the spatial frequency α of the resulting fringe pattern can be determined from

$$2\pi\alpha\xi = k\xi[\sin(\theta_a) + \sin(\theta_u)]. \quad (2.33)$$

Using $k = \frac{2\pi}{\lambda}$ and solving for α in Eq. (2.33) gives the resulting formula

$$\alpha = \frac{\sin(\theta_a) + \sin(\theta_u)}{\lambda}. \quad (2.34)$$

Note that for a given total angle between the reference and object waves, the spatial frequency α can still vary. For example, let $\theta_a = 30^\circ$ and $\theta_u = 60^\circ$. Then

$\alpha = \frac{1.366}{\lambda}$. However, if $\theta_a = 45^\circ$ and $\theta_u = 45^\circ$, then $\alpha = \frac{1.414}{\lambda}$. So, even though $\theta_a + \theta_u = 90^\circ$ in both cases, α is different for each case. It can be shown that, for a given sum $\theta_a + \theta_u$, the maximum α occurs when $\theta_a = \theta_u$. Figure 2.7 illustrates the relation between α , λ , θ_a , and θ_u .

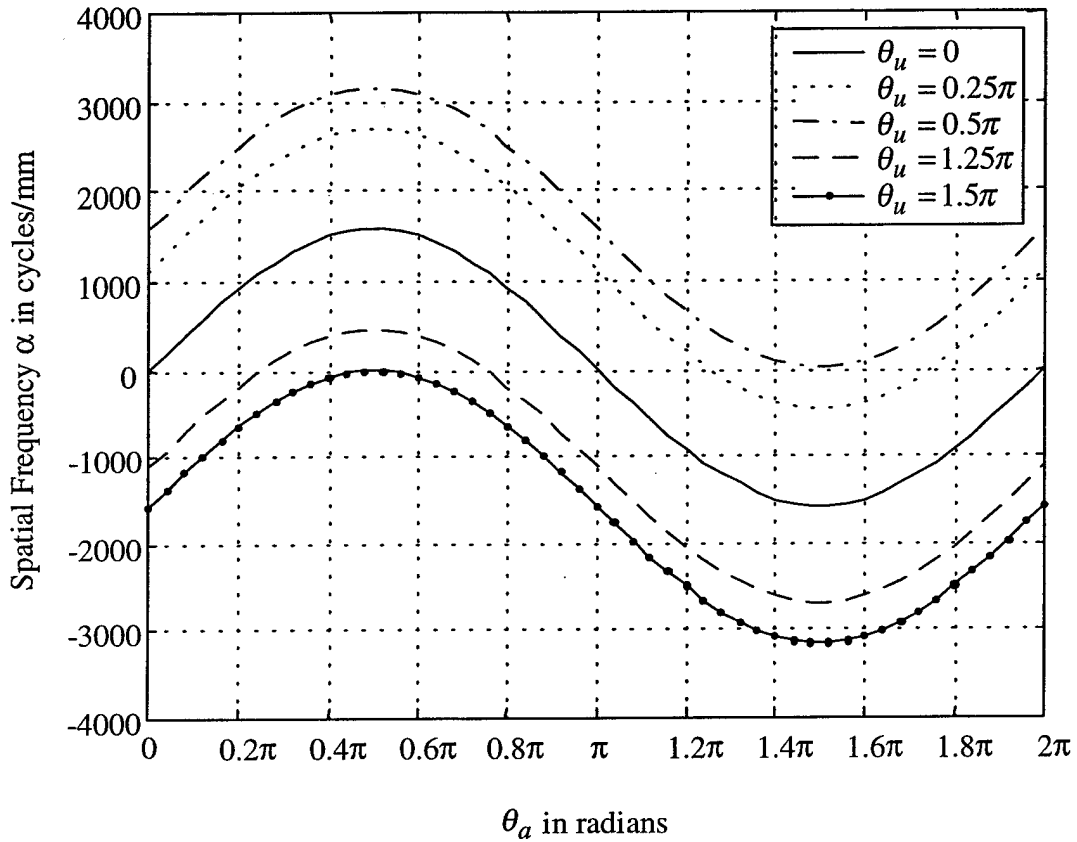


Figure 2.7 Plot of spatial frequencies resulting from interference between various reference plane wave angles θ_a and object angles θ_u for $\lambda=632.8\text{nm}$.

3.0 DIGITAL RECONSTRUCTION

Digital reconstruction of holograms has several advantages. One is that it avoids the speckle that occurs during the optical reconstruction (but not the speckle recorded on the hologram). Another advantage is that the raw data recorded on the hologram is utilized and so allows more direct reconstruction of the image. Another advantage is that the depth of field is large which is good for reconstruction as in tomographic back-projection.

Of course, digital reconstruction has its disadvantages as well. Some of its limitations are addressed in this chapter. For instance, the large depths of field are a result of low resolution. Each of the steps shown in Fig. 1.2 has an effect on the quality of the reconstructed image.

3.1 Digitization of the Hologram

3.1.1 Microscope Objectives and Numerical Aperture

In order to scan a hologram into a computer, the instrument must resolve the very small fringes that are recorded on the hologram. Typically, it is necessary to utilize a microscope. So, the attention is turned to the ability of microscopes, or more appropriately, the microscope objective to resolve the fringes.

The mathematical development starts with the Fraunhofer approximation for the field amplitude given in Eq. (2.11). The integral in that equation is carried out over the aperture of the system. For the case of a microscope objective, the aperture is circular. Using a circular aperture as the limits of integration and assuming a plane wave incident upon this aperture, Eq. (2.11) develops into a Fourier-Bessel transform. The solution of this transform equation is well known and the resulting intensity described is known as the Airy pattern [Hecht, 1998, p.459-461].

The Airy pattern is the physical phenomenon that is used to establish the resolution limit of an optical system. When this limitation applies, the optical system is said to be diffraction-limited. Depending on the quality of the optical components in the system, the resolving power may be considerably worse than the idealized diffraction-limited case.

Assuming a diffraction-limited optical system, a quantifiable measurement of the resolving power of a system is needed. One method used is the Rayleigh criterion. This criterion states that two points can be resolved if the maximum of one point's Airy pattern overlaps at the first zero of the other point's Airy pattern [Longhurst, 1967, p. 302]. This is illustrated in Fig 3.1. From the superposition principle, the Airy pattern of each point adds to give the result. In Fig 3.1, there is a line that droops down between the peaks of the individual Airy patterns. That droop indicates that the individual points of light can still be distinguished.

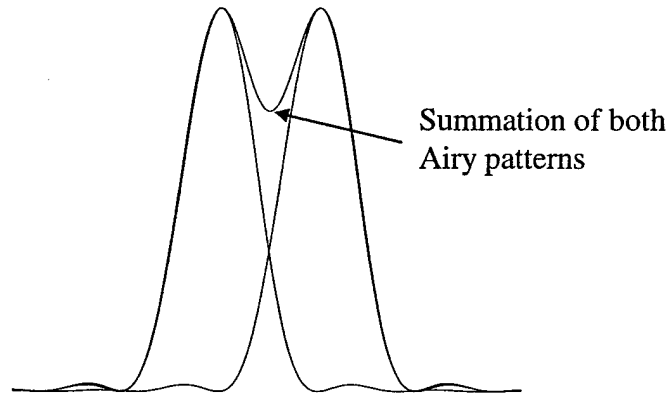


Figure 3.1 Illustration of the Rayleigh criterion.

Using the Rayleigh criterion, the limit of resolution is then the radius to the first dark ring. This radius is given by

$$h' = \frac{0.61\lambda'}{\sin \theta'}, \quad (3.1)$$

where λ is the wavelength in the image space and θ' is the semi-angle of the emergent cone of light [Longhurst, 1967, p. 220-223]. Since

$$\lambda' = \frac{\lambda_o}{n'}, \quad (3.2)$$

where λ_o is the wavelength in a vacuum and n is the refractive index of the image space [Hecht, 1998, p. 103], then Eq. (3.1) can be written as

$$h' = \frac{0.61\lambda_o}{n' \sin \theta'} \quad (3.3)$$

[Longhurst, 1967, p. 224]. It should be noted that Longhurst [1967, p. 224] used λ instead of λ_o to represent the wavelength in a vacuum.

The Rayleigh criterion is perhaps the most commonly used criterion. However, at least one other deserves mention. It is called the Sparrow criterion. The Sparrow criterion defines the resolution limit as occurring when the intensity sums result in a flat top or plateau between the two peaks versus the droop shown in Fig 3.1 [Reynolds, et. al., 1989, p. 40-41]. It can be inferred from Reynolds, et. al. [1989, p. 42] that the Sparrow criterion gives a resolution limit of

$$h' = \frac{0.4736\lambda_o}{n' \sin \theta'}. \quad (3.4)$$

When using a microscope (as in our situation of scanning the fringes from holographic film), the resolving power requires additional development. Figure 3.2 depicts the setup for a microscope. In this case, the optical sine theorem should be applied to relate the image points to the object points. Using Fig. 3.2, this relation is given by

$$nh \sin \theta = n'h' \sin \theta', \quad (3.5)$$

where n and n' are the indices of refraction of the object space and image space, respectively [Hecht, 1998, p. 264], [Longhurst, 1967, p. 297-298]. Equations (3.3) and (3.5) can be combined to give the distance h between the object points at the diffraction limit to be

$$h = \frac{0.61\lambda_o}{n \sin \theta} \quad (3.6)$$

[Longhurst, 1967, p. 303]. The quantity $n \sin \theta$ is defined as the numerical aperture and denoted by NA [Longhurst, 1967, p. 303]. Substituting this into Eq. (3.6) gives the resolution limit in the object plane for a diffraction-limited system as

$$h = \frac{0.61\lambda_o}{NA}. \quad (3.7)$$

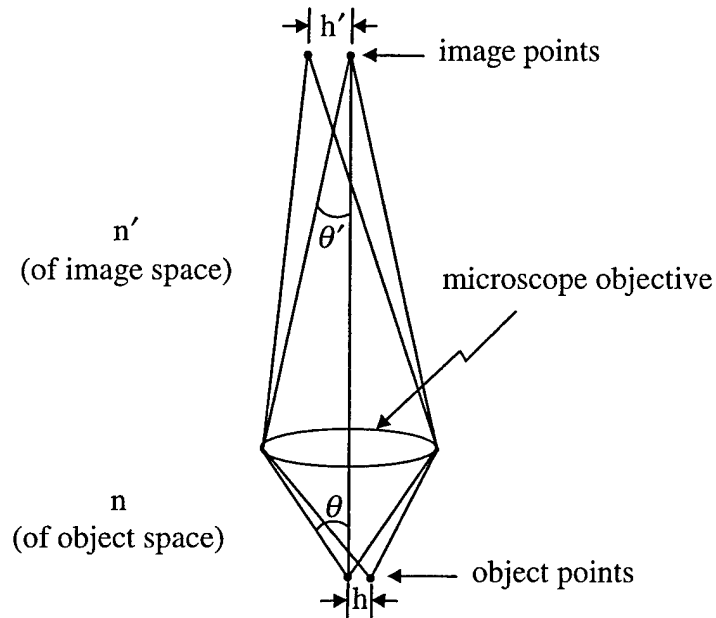


Figure 3.2 Illustration for the Optical Sine Theorem.

3.1.2 Finite Pixel Size

Ideally, sampling of the hologram for digitization is done with perfect impulses. However, this cannot be done with real components. The finite width of the true sampling function will affect the reconstructed image. This section will explore the effect.

After the hologram is magnified with a microscope of appropriate resolving power, the next step is to use some device to convert the optical image of the hologram into an electronic form that can be stored and processed in a computer. A typical device used is an electronic camera. The component in the camera that does the actual conversion from photonic to electronic energy is referred to as an imager. The most common imager used is the charge-coupled device or CCD. The CCD consists of an

array of pixels, each of which provides a packet of charge to the camera, which converts the charge into a voltage signal. The amount of charge from each pixel depends on a number of factors including the amount of photon energy it receives, the time of exposure, and the size of the pixel. It is the size of the pixels, the spacing between pixels, and the effect they have on the image quality that now will be addressed.

The charge output from each pixel is the result of accumulating photon energy over its area and converting the sum of this energy into one charge packet. Thus, the output is actually an integration of the energy over the pixel area. We make the assumption that the pixels are square and denote the length of each side as W . Also, the pixel-to-pixel spacing in the ξ and η dimensions are defined as ξ_s and η_s , respectively. Figure 3.3 illustrates these dimensions. Then in mathematical terms, the charge from the m th pixel is

$$S_{m,n} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U(\xi, \eta) \text{rect}\left(\frac{\xi - m\xi_s}{W}\right) \text{rect}\left(\frac{\eta - n\eta_s}{W}\right) d\xi d\eta. \quad (3.8)$$

where the function $\text{rect}(\cdot)$ is defined by [Kreis, 1996, p. 268]

$$\text{rect}(x) = \begin{cases} 1, & |x| < \frac{1}{2} \\ 0, & \text{elsewhere.} \end{cases} \quad (3.9)$$

When the charge from all pixels in the CCD are put together, the result is a sampling of the image described by

$$\hat{U}_s(\xi, \eta) = \sum_m \sum_n \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U(\xi, \eta) \text{rect}\left(\frac{\xi - m\xi_s}{W}\right) \text{rect}\left(\frac{\eta - n\eta_s}{W}\right) d\xi d\eta \cdot \delta(\xi - m\xi_s) \delta(\eta - n\eta_s). \quad (3.10)$$

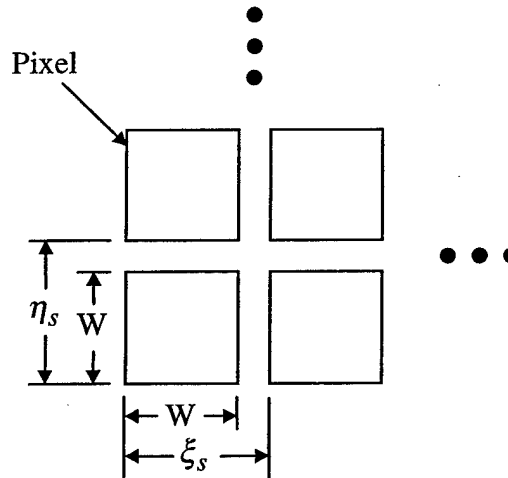


Figure 3.5 Pixel Size and Spacing.

It will now be shown that Eq. (3.10) is equivalent to the ideal sampling of the convolution of the image with a *rect* function of dimensions the same as a pixel. The continuous convolution [Ziemer, et. al., 1983, p. 44] of the entire image with a 2D *rect* function with a width of W in each direction is

$$\hat{U}_c(\xi, \eta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U(a, b) \text{rect}\left(\frac{\xi - a}{W}\right) \text{rect}\left(\frac{\eta - b}{W}\right) da db. \quad (3.11)$$

Then, if the image described by Eq.(3.11) is perfectly sampled, the result can be described by

$$\hat{U}_{cs}(\xi, \eta) = \sum_m \sum_n \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U(a, b) \text{rect}\left(\frac{\xi - a}{W}\right) \text{rect}\left(\frac{\eta - b}{W}\right) da db \delta(\xi - m\xi_s) \delta(\eta - n\eta_s). \quad (3.12)$$

The impulse functions in Eq. (3.12) are not a function of the integration variable and can be brought inside the integral. This gives

$$\hat{U}_{cs}(\xi, \eta) = \sum_m \sum_n \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U(a, b) \text{rect}\left(\frac{\xi - a}{W}\right) \text{rect}\left(\frac{\eta - b}{W}\right) \delta(\xi - m\xi_s) \delta(\eta - n\eta_s) da db. \quad (3.13)$$

Using the fact that [Ziemer, et. al., 1983, p. 21]

$$x(t) \delta(t - t_o) = x(t_o) \delta(t - t_o) \quad (3.14)$$

results in

$$\begin{aligned} \hat{U}_{cs}(\xi, \eta) = \sum_m \sum_n \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U(a, b) \text{rect}\left(\frac{m\xi_s - a}{W}\right) \text{rect}\left(\frac{n\eta_s - b}{W}\right) da db \\ \cdot \delta(\xi - m\xi_s) \delta(\eta - n\eta_s). \end{aligned} \quad (3.15)$$

The double integral of Eq. (3.15) can be recognized as a convolution. Since *rect* is an even function, Eq. (3.15) can also be written as

$$\begin{aligned} \hat{U}_{cs}(\xi, \eta) = \sum_m \sum_n \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U(a, b) \text{rect}\left(\frac{a - m\xi_s}{W}\right) \text{rect}\left(\frac{b - n\eta_s}{W}\right) da db \\ \cdot \delta(\xi - m\xi_s) \delta(\eta - n\eta_s). \end{aligned} \quad (3.16)$$

Comparison of Eqs. (3.10) and (3.16) shows that they are equal. Therefore, it can be concluded that the finite pixel width and its operation as an integrator has the same effect as convolution of the entire image continuously with a *rect* function the size of a pixel followed by perfect sampling.

One property of the Fourier transform is that convolution in one domain is the equivalent of multiplication in the other domain [Ziemer, et. al., 1983, p. 132]. Therefore

$$\begin{aligned} U(a, b) * \left[\text{rect}\left(\frac{m\xi_s - a}{W}\right) \text{rect}\left(\frac{n\eta_s - b}{W}\right) \right] \\ \leftrightarrow \mathfrak{F}[U(a, b)] \mathfrak{F}\left[\text{rect}\left(\frac{m\xi_s - a}{W}\right) \text{rect}\left(\frac{n\eta_s - b}{W}\right) \right] \end{aligned} \quad (3.17)$$

where $*$ indicates convolution, \leftrightarrow indicates a Fourier transform pair, and \mathfrak{F} indicates the Fourier transform. The Fourier transform of the *rect* function is

$$\mathfrak{F}\left[\text{rect}\left(\frac{m\xi_s - a}{W}\right) \right] = W \text{sinc}(fW). \quad (3.18)$$

Assuming the spectral extent of $U(a,b)$ to be $\pm F_u$, then the Fourier transform of Eq. (3.15) is as illustrated in Fig. 3.4. Figure 3.4b illustrates the special case where the pixel width W and the pixel spacing ξ_s are equal. For this case, the amplitude reduction at $\frac{f_s}{2}$ due to the *sinc* function is given by

$$W \text{sinc} fW = \frac{W \sin(\pi fW)}{\pi fW} \bigg|_{f=\frac{f_s}{2}} = \frac{W \sin\left[\left(\pi\right)\left(\frac{f_s}{2}\right)\left(\frac{1}{f_s}\right)\right]}{\left(\pi\right)\left(\frac{f_s}{2}\right)\left(\frac{1}{f_s}\right)} = \frac{2W}{\pi}. \quad (3.19)$$

This corresponds to a 1.96 dB reduction in frequency amplitude for frequencies at half the sampling frequency.

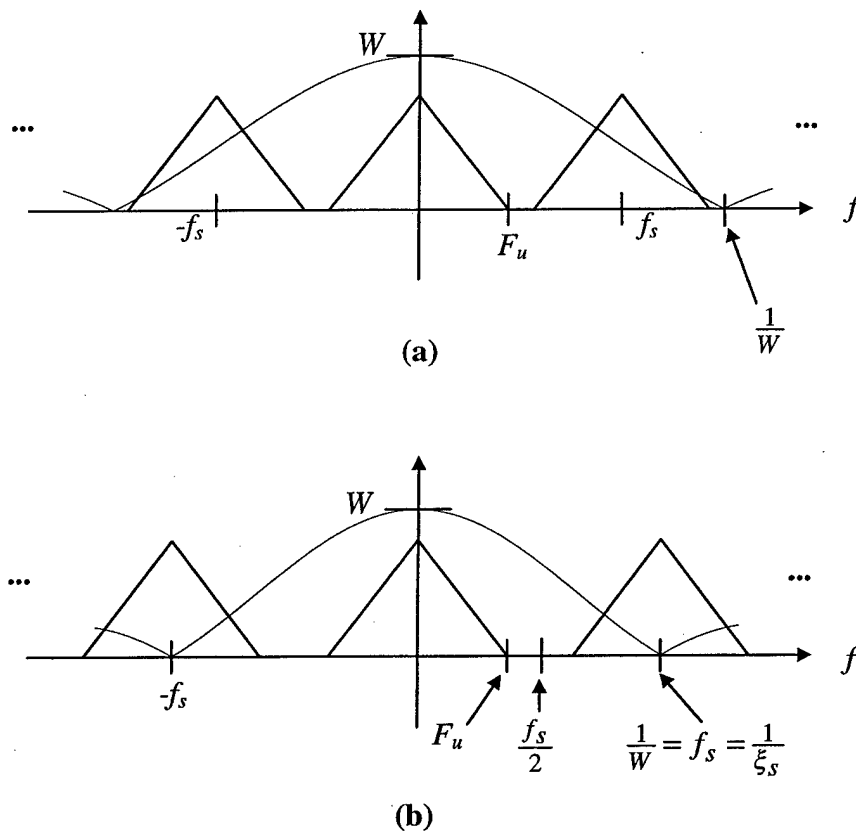


Figure 3.4 Frequency spectrum of image convolved with a pixel of finite width and perfectly sampled: (a) General case. (b) Special case where the pixel width and pixel spacing are equal.

3.2 Effect of Limited Scan Area

The width of the pixels is not the only effect imposed by an electronic camera. Another consideration is the number of pixels in each dimension. The digitized hologram

obtained from a camera will be comprised of this number of pixels. As a result, only a small portion of the whole hologram is digitized. The effect of this is examined below.

3.2.1 Maximum Theoretical Resolution

Figure 3.5 illustrates the coordinate system for the object, hologram, and image planes. After some mathematical development, the wavefront at the observation region (the hologram plane) was determined to be

$$\mathbf{U}(\xi, \eta) = \frac{e^{jk d}}{j \lambda d} \exp \left[\frac{jk}{2d} (\xi^2 + \eta^2) \right] \iint_{\Sigma} \mathbf{O}(x, y) \exp \left[\frac{jk}{2d} (x^2 + y^2) \right] \exp \left[-\frac{j2\pi}{\lambda d} (x\xi + y\eta) \right] dx dy. \quad (3.20)$$

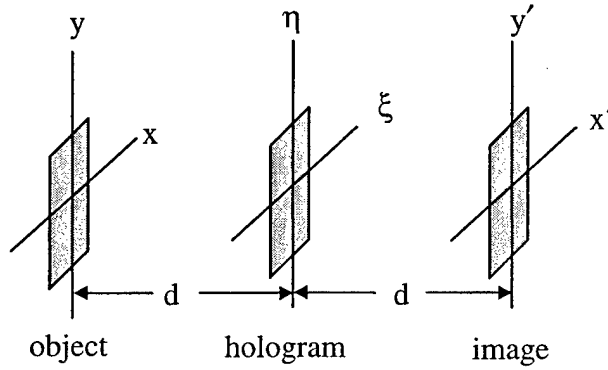


Figure 3.5 Coordinate system for the object, hologram, and image planes.

The goal is to recreate the wavefront from the object, $\mathbf{O}(x, y)$. The integral is just the two dimensional forward Fourier transform of

$$\mathbf{O}(x, y) \exp \left[\frac{jk(x^2 + y^2)}{2d} \right], \quad (3.21)$$

except that the λd in the last term of (3.20) is not in the transform definition. This must be taken into account for proper scaling in the observation plane. This can be accounted for through use of the scale change theorem of Fourier transforms. The scale changes made are

$$p = \frac{\xi}{\lambda d} \quad (3.22)$$

and

$$q = \frac{\eta}{\lambda d}. \quad (3.23)$$

Substituting these into Eq. (3.20) gives

$$\begin{aligned}
U(p\lambda d, q\lambda d) &= \frac{e^{jkd}}{j\lambda d} \exp\left[\frac{jk}{2d}((p\lambda d)^2 + (q\lambda d)^2)\right] \\
&\cdot \iint_{\Sigma} O(x, y) \exp\left[\frac{jk}{2d}(x^2 + y^2)\right] \exp[-j2\pi(xp + yq)] dx dy \\
&= \frac{e^{jkd}}{j\lambda d} \exp\left[\frac{jk}{2d}((p\lambda d)^2 + (q\lambda d)^2)\right] \mathfrak{F}\left\{O(x, y) \exp\left[\frac{jk}{2d}(x^2 + y^2)\right]\right\}
\end{aligned} \tag{3.24}$$

where $\mathfrak{F}(\bullet)$ indicates the forward Fourier transform. The substitutions reveal that p and q are the frequency domain variables of x and y .

Considering the ξ dimension, if the scan is limited to $\pm \frac{\xi_{\max}}{2}$ and is sampled N times, then the sample size is

$$\Delta\xi = \frac{\xi_{\max}}{N}, \tag{3.25}$$

which in the p dimension is

$$\Delta p = \frac{\Delta\xi}{\lambda d}. \tag{3.26}$$

The highest spatial frequency that can be resolved is then [Strum, Kirk, 1989, p. 391]

$$\alpha = \Delta p \left(\frac{N}{2}\right) = \frac{\Delta\xi N}{2\lambda d} = \frac{\xi_{\max}}{2\lambda d}. \tag{3.27}$$

For $L = \xi_{\max}$, the maximum resolution is

$$h_{lp} = \frac{1}{\alpha} = \frac{2\lambda d}{L}, \tag{3.28}$$

where the subscript lp is used to indicate the resolution is in terms of line pairs.

Therefore, the resolution in terms of lines is

$$h = \frac{h_{lp}}{2} = \frac{1}{2\alpha} = \frac{\lambda d}{L}. \tag{3.29}$$

For example, if $\lambda=632.8\text{nm}$, $d=50\text{mm}$, and $L=0.5\text{mm}$, then

$$h = \frac{\lambda d}{L} = \frac{(632.8\text{nm})(150\text{mm})}{0.5\text{mm}} = 189.8\mu\text{m}. \tag{3.30}$$

Thus, the highest spatial frequency that can be resolved is

$$\alpha = \frac{1}{2h} = \frac{1}{2(189.8\mu\text{m})} = 2.634 \frac{lp}{\text{mm}}. \tag{3.31}$$

3.2.2 Mosaicking

From the preceding development on the maximum resolution obtainable from a digitized portion of a hologram, it is clear that larger scan areas are needed to improve the resolution of reconstructing holograms. One approach is to sample overlapping images from the hologram and compose these individual images into a larger image. This

process is commonly referred to as image mosaicking. Obviously, accurate placement of the individual images into the correct location is needed to avoid errors in the reconstruction.

To examine the impact of inaccurate placement, a method for measuring the effects must be chosen. There are a number of methods used to qualitatively and quantitatively measure optical systems [Mouroulis, Zhang, 1992]. One of the more commonly used is the radius of encircled energy [Rivolta, 1986, p. 2404]. This is the technique chosen for evaluating the accuracy needed for mosaicking small images together to form a larger image.

The encircled energy of radius R is defined as the fraction of the total energy of the point spread function contained within the radius R from the center of the diffraction pattern [Slepian, 1965, p. 1111], [Clark, et. al., 1984]. A commonly used figure of merit is the radius that encircles 84% of the energy. This is the fraction of energy contained within the Airy disk in the diffraction-limited case [Mouroulis, Zhang, 1992]. However, as will be seen later, selecting a radius was not needed.

Having chosen the encircled energy as the method to analyze mosaicking, this method must now be applied in some fashion to obtain empirical results. Analysis was performed through simulation. The program code written to perform the simulation is provided in Appendix A. Comments are provided within the code to help in understanding its function. A description of the approach used in the simulation follows.

The approach used in the simulation was to analyze the effects of mosaicking error on sine waves. A perfect mosaic would consist of several segments of the sine wave matched together to form one long sine wave. However, if any error occurs in the mosaicking process, then there would be a discontinuity at the location where any two mismatched segments were joined together. Figure 3.6 illustrates the two situations.

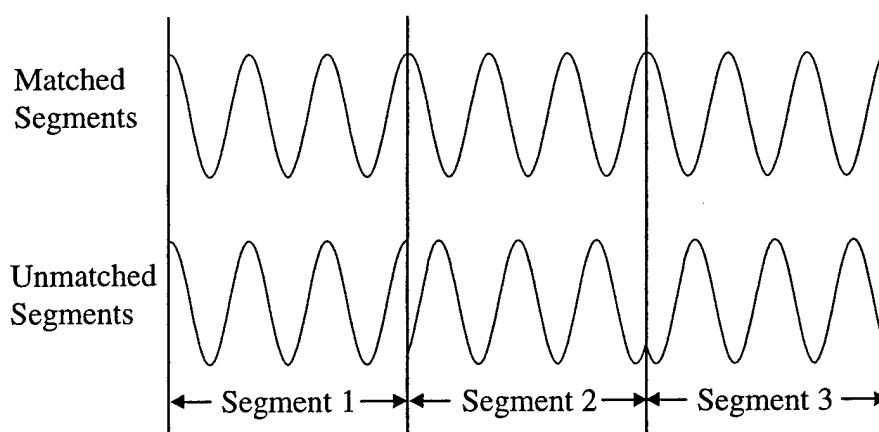


Figure 3.6 Mosaicking with and without error.

Mathematically, each discontinuity can be modeled as a change in phase. Also, the phase change at each successive discontinuity is relatively to the previous phase change. Suppose the second segment is mismatched with respect to the first by $\Delta\phi_1$ and that the third segment is perfectly matched with the second segment. The result is that the third segment must be in error by the same phase amount $\Delta\phi_1$ with respect to the first segment. Therefore, if the third segment is mismatched by $\Delta\phi_2$ with respect to the second, then its mismatch relative to the first segment is $\Delta\phi_1 + \Delta\phi_2$.

The simulation used the above described accumulation of phase from segment to segment to represent mosaicking error in the analysis. Each segment was given a uniformly random phase deviation bound by $\pm\Delta\phi$ with respect to the previous segment. Thus, the actual phase deviation of each segment from the first was an integer multiple of $\pm\Delta\phi$. For instance, the second segment would deviate from the first by $\pm\Delta\phi$ and the third would deviate by $\pm2\Delta\phi$ with respect to the first segment, and so on.

Once the phase for each segment is determined, the composite signal would be generated. Then, a fast Fourier transform (FFT) is performed on the composite signal. The result is then squared to obtain data representing energy. The form of this result is a *sinc* function. This is due to the finite size of the signal. Of course, if the signal were infinite, the result would be impulses at \pm the frequency f of the sinewave. Next, the encircled energy is computed by summing the values at increasingly larger distances from the ideal frequency f . Finally, the whole process is repeated many times and averaged for statistical purposes.

Data from the simulations are graphed in Figs. 3.7 to 3.16. There are 6 curves in each graph. The maximum phase deviation for each curve is $0, \pm\frac{\pi}{4}, \pm\frac{\pi}{2}, \pm\frac{3\pi}{4}, \pm\pi, \pm\frac{5\pi}{4}$ starting with the top curve. The dashed curve in each graph is the ideal encircled energy curve for one segment. If mosaicking is to provide improvement, then the encircled energy curve of the mosaicked signal should lie above that of one segment. From these graphs, one can see an interesting result. As the allowed phase deviation is increased, the curves gradually approach the ideal one segment curve and coincide with it when the maximum phase deviation is $\pm\pi$. This is true in every case, regardless of the segment size, frequency of the signal, or number of segments. As the maximum phase deviation is increased further, the curves drop below the ideal curve but then turn back and eventually rise above the ideal. This cycle apparently continues for even larger phase deviations.

The implications of this result are important. The worst case scenario in terms of error tolerance is when the signal (or image) contains frequencies up to the maximum allowed of half the sampling frequency. In this case, one pixel would represent a phase shift of π . Thus, the error must be kept within one pixel. However, the π phase shift corresponds to an encircled energy curve equal to that of one segment. The purpose of mosaicking is to improve the resolution capability. Therefore, the accuracy must be

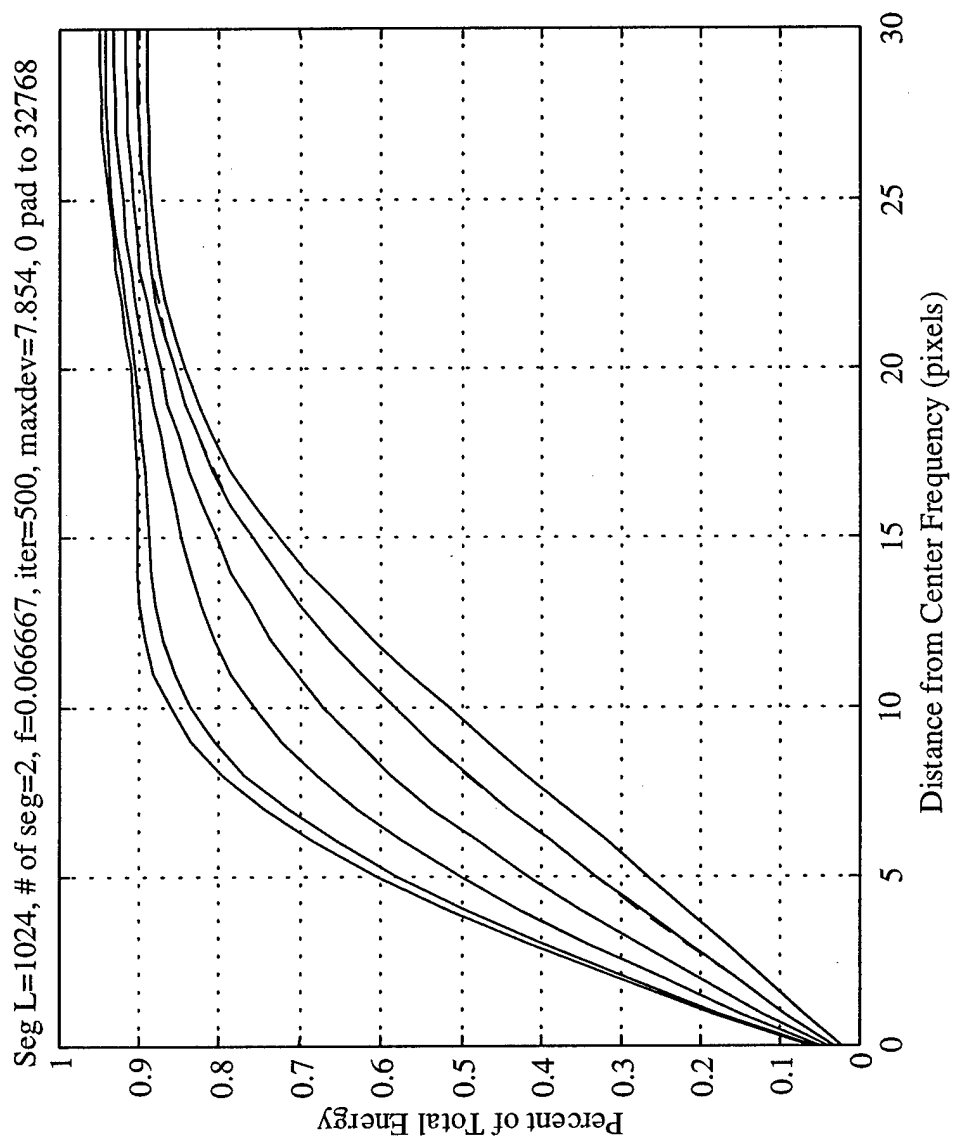


Figure 3.7 Encircled energy plots for 2 segments of 1024 pixels each and a frequency of 1/15.

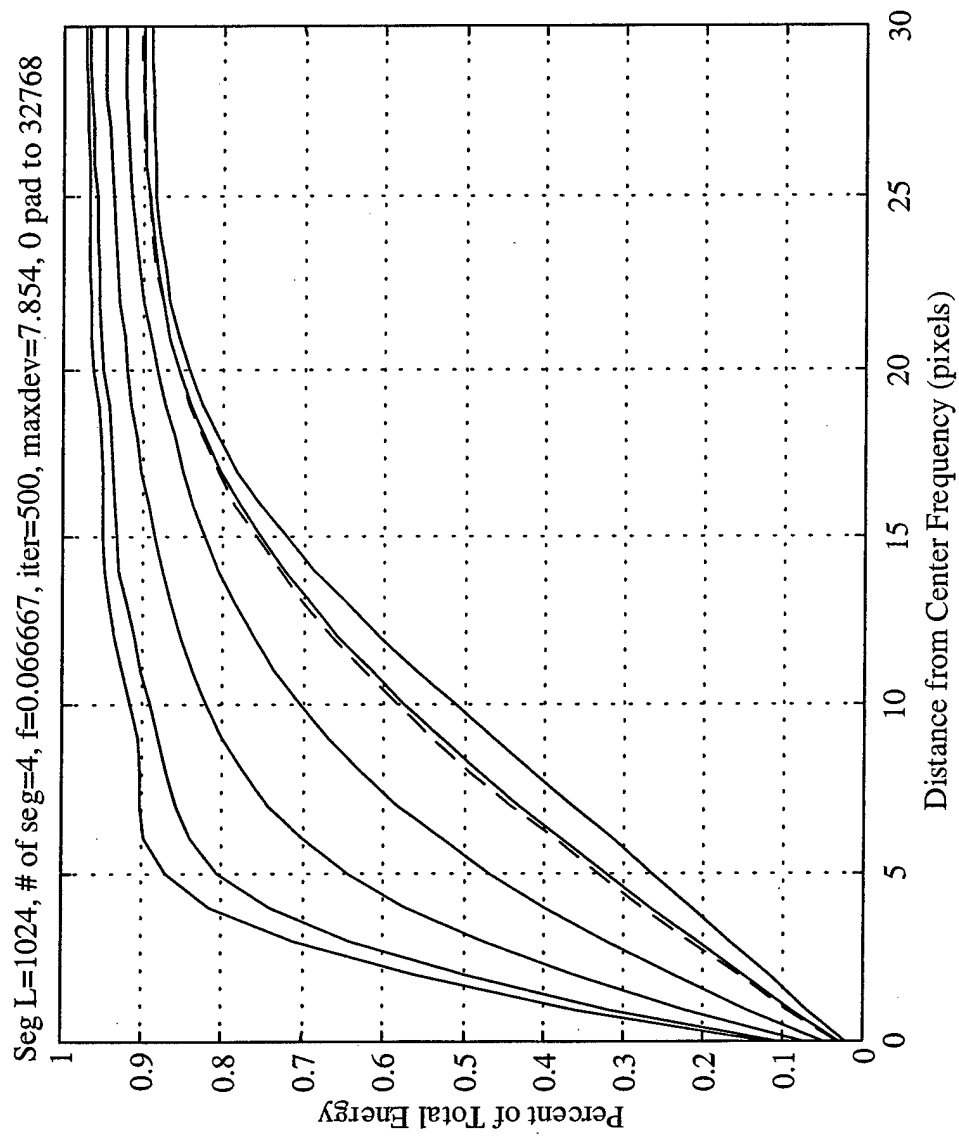


Figure 3.8 Encircled energy plots for 4 segments of 1024 pixels each and a frequency of 1/15.

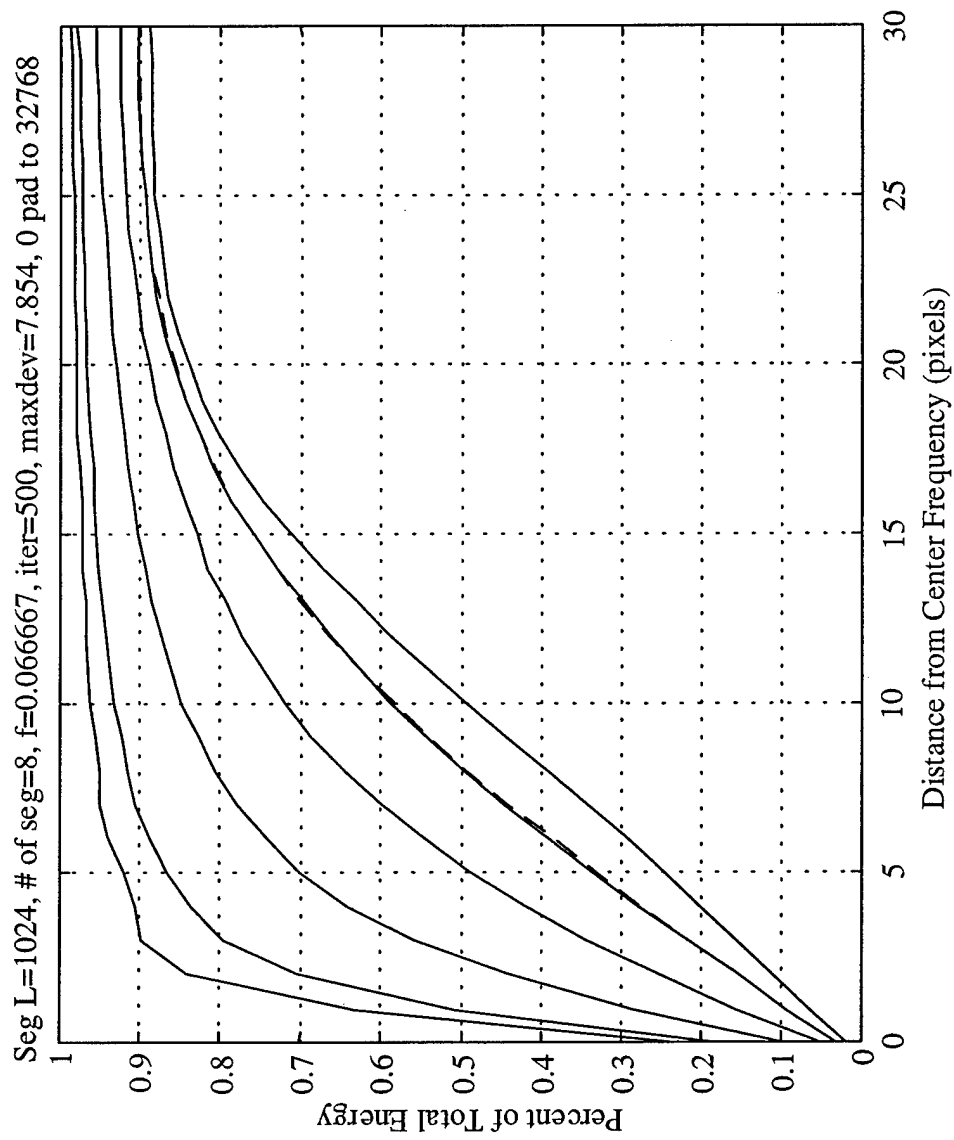


Figure 3.9 Encircled energy plots for 8 segments of 1024 pixels each and a frequency of 1/15.

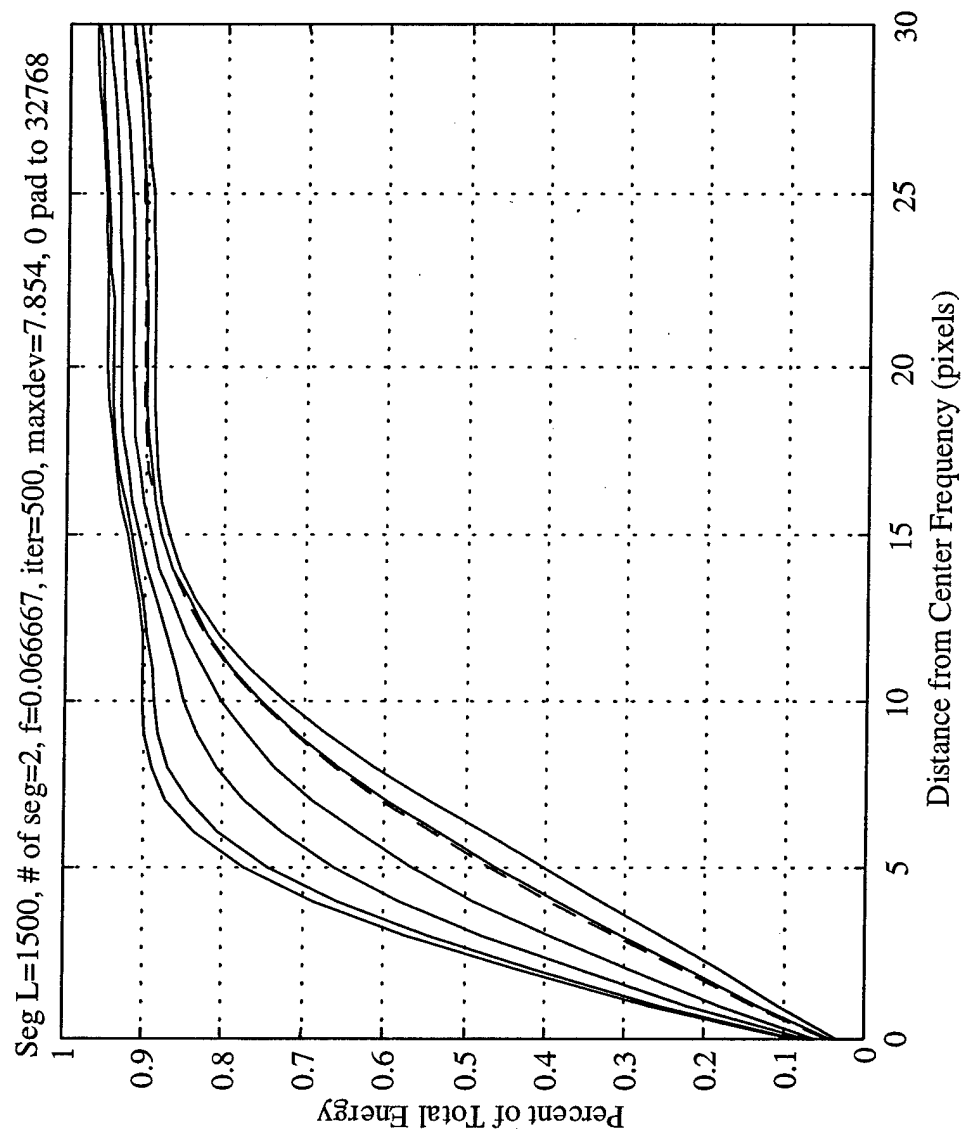


Figure 3.10 Encircled energy plots for 2 segments of 1500 pixels each and a frequency of 1/15.

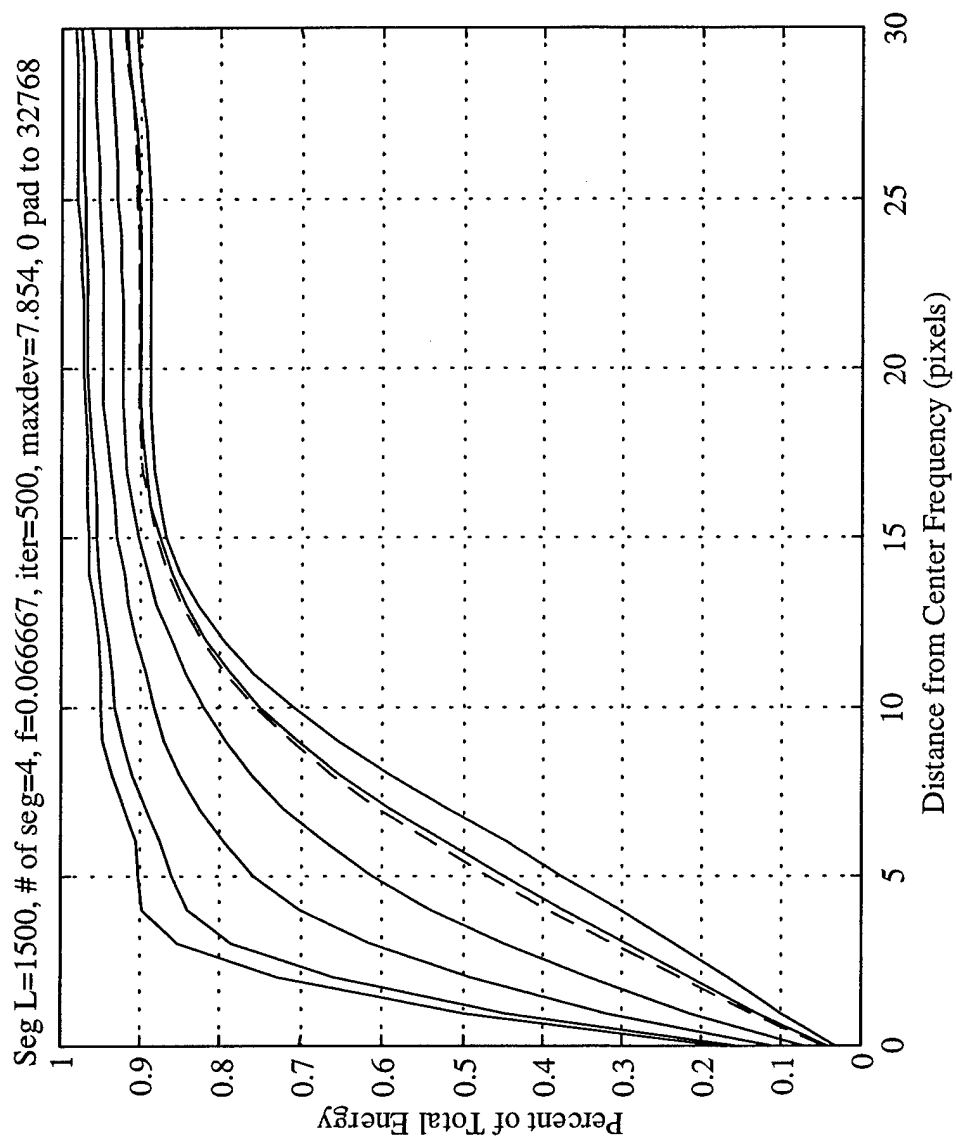


Figure 3.11 Encircled energy plots for 4 segments of 1500 pixels each and a frequency of 1/15.

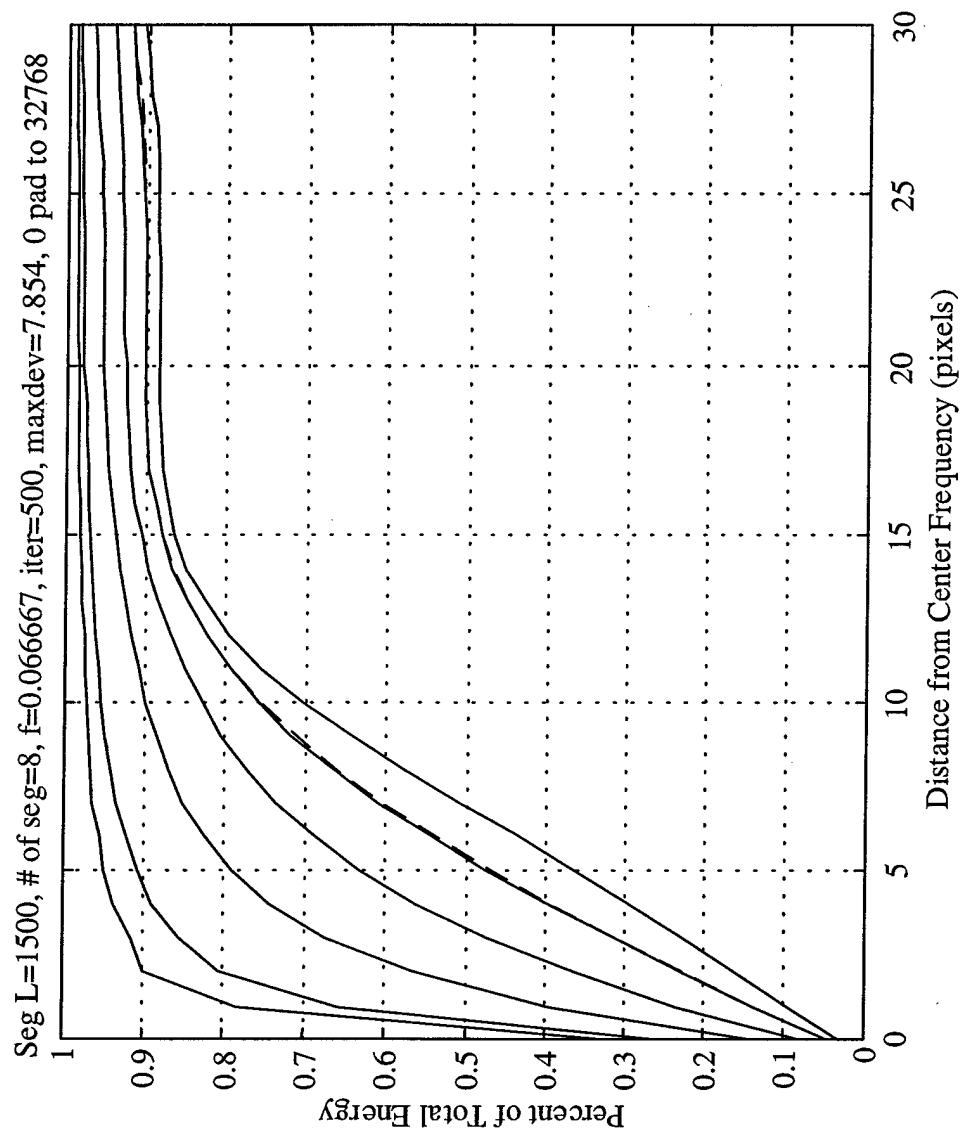


Figure 3.12 Encircled energy plots for 8 segments of 1500 pixels each and a frequency of 1/15.

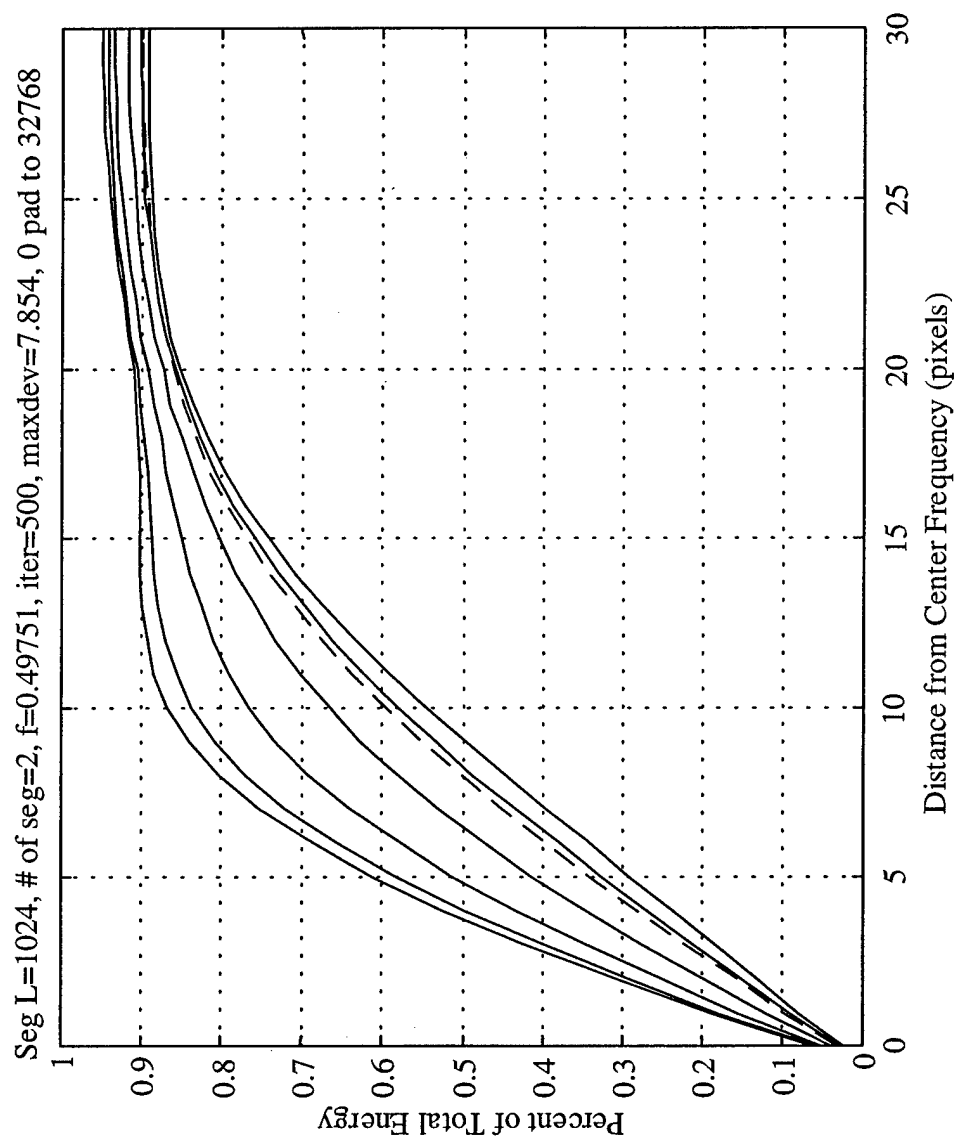


Figure 3.13 Encircled energy plots for 2 segments of 1024 pixels each and a frequency of $1/2.01$.

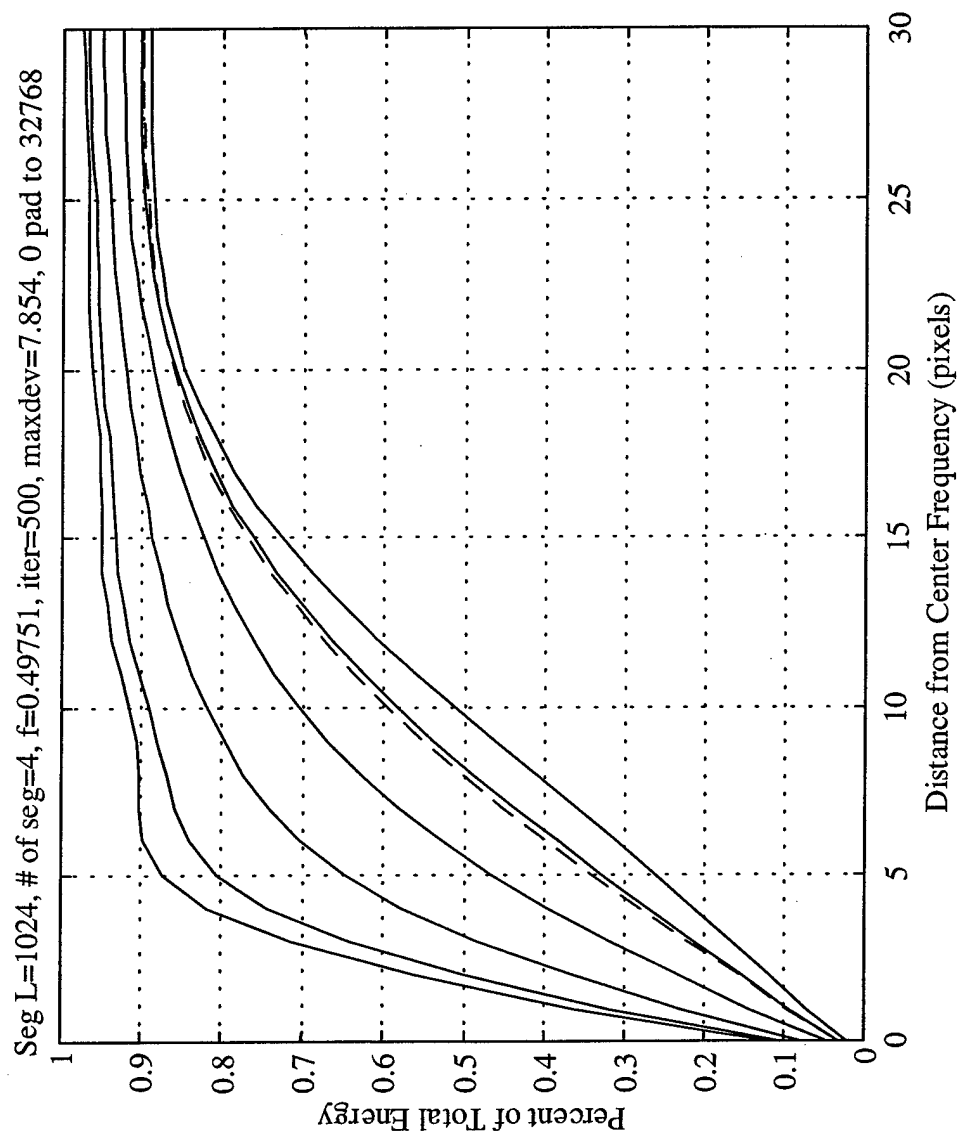


Figure 3.14 Encircled energy plots for 4 segments of 1024 pixels each and a frequency of $1/2.01$.

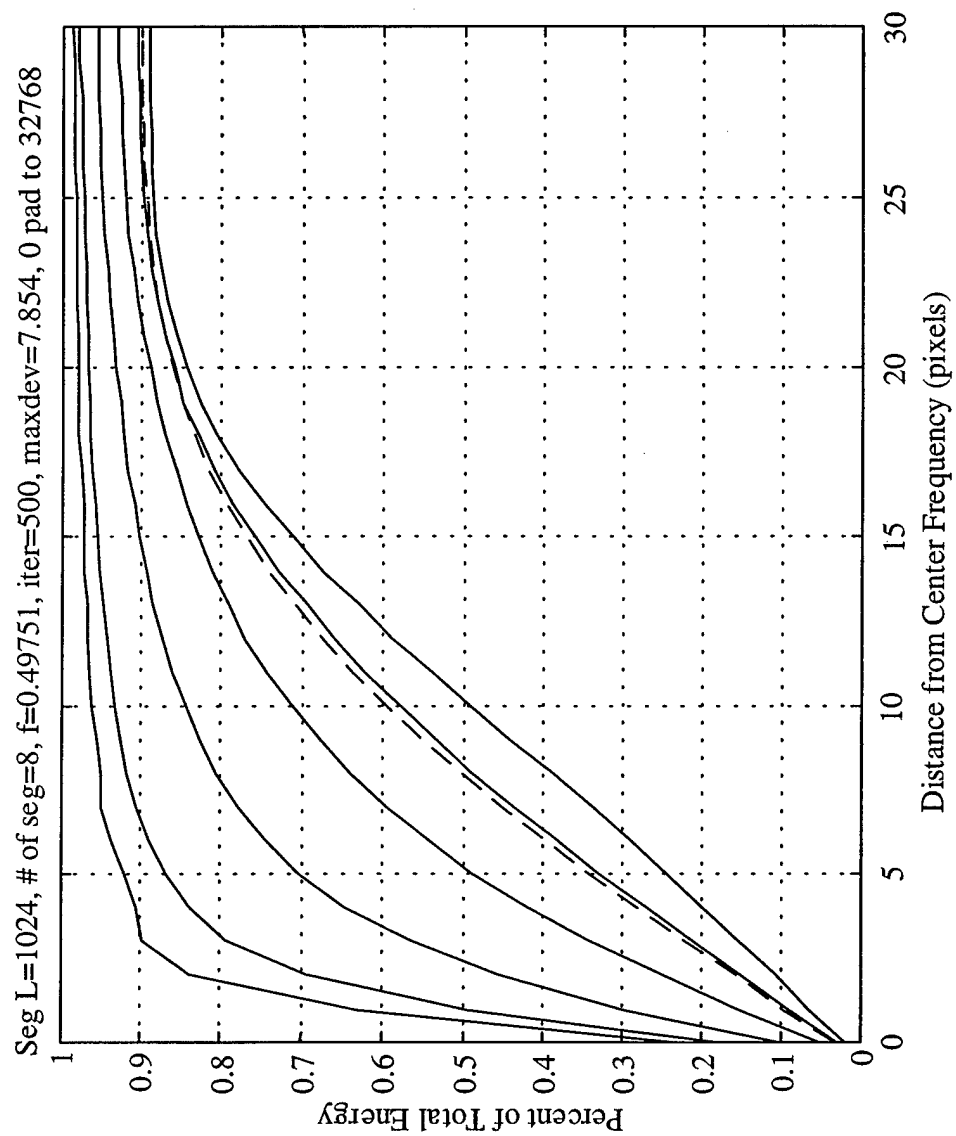


Figure 3.15 Encircled energy plots for 8 segments of 1024 pixels each and a frequency of $1/2.01$.

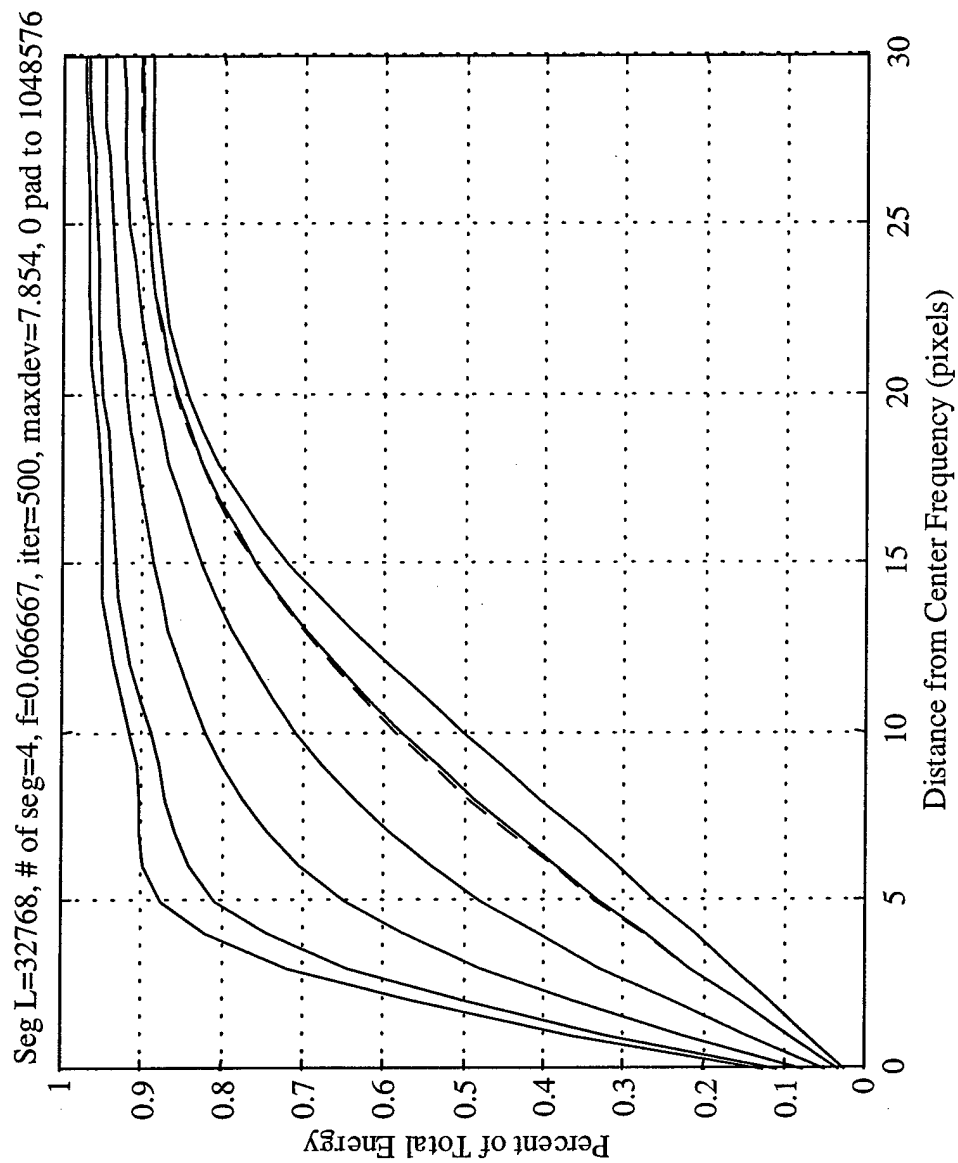


Figure 3.16 Encircled energy plots for 4 segments of 32768 pixels each and a frequency of 1/2.01.

better than the π phase shift. If $\frac{\pi}{2}$ is selected as the needed phase accuracy in placement, then, for the case of frequencies at the Nyquist limit, this means that the accuracy must be better than one pixel. For more error tolerance, the maximum frequency in the image must be even less than the Nyquist limitation of half the sampling frequency.

A mosaicking program was written as part of this research that would do simple placement of images utilizing a basic correlation technique to find the basic match among images. Once the mosaicking accuracy requirement became clearer, further development of this program was abandoned. The code for this program is provided in Appendix B.

3.2.3 Depth of Field

Depth of field is the amount of movement of the object plane allowed before the stationary image plane shows deterioration of the image. Similarly, depth of focus is the amount of movement of the image plane allowed before the stationary appears out of focus [Longhurst, 1967, p. 306-308]. If depth of focus can be defined, then the depth of field can be found through use of the thin lens equation. Figure 3.17 illustrates the relationships of the various parameters.

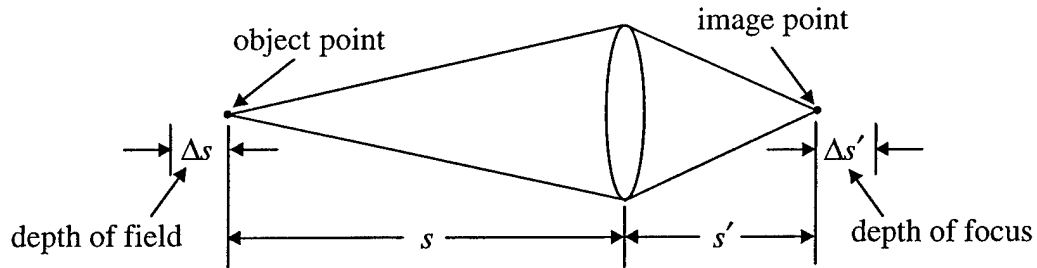


Figure 3.17 Depth of field geometry.

The depth of focus in an optical system is closely related to its resolution ability. Just as the Rayleigh criterion provides a measure of resolution, there is a similar measure along the axis of the optical system. Essentially, the light distribution at the focal point must be considered in three dimensions instead of two [Born, Wolf, 1980, p. 435]. The intensity along the axis for a circular aperture is of the form [Born, Wolf, 1980, p. 441]

$$I = \left(\frac{\sin u/4}{u/4} \right)^2 I_o, \quad (3.32)$$

where [Born, Wolf, 1980, p. 437]

$$u = \frac{2\pi}{\lambda} \left(\frac{r}{R} \right)^2 z'. \quad (3.33)$$

In Eq. (3.33), z' is the optical axis, r is the radius of the aperture, and R is the radius of the wavefront at the aperture [Born, Wolf, 1980, p. 435-436]. An intensity loss of 20%

of that at the center is considered acceptable [Born, Wolf, 1980, p. 441]. This corresponds to

$$u \cong 3.2. \quad (3.34)$$

Substituting Eq. (3.34) into Eq. (3.33) gives the depth of focus as [Born, Wolf, 1980, p. 441]

$$\Delta s' \cong \pm \frac{1}{2} \left(\frac{R}{r} \right)^2 \lambda. \quad (3.35)$$

This is usually combined with the Rayleigh criterion of resolution, which can be stated as [Hecht, 1998, p. 464]

$$h_R = \frac{1.22 f \lambda}{2r}, \quad (3.36)$$

to give the depth of focus as

$$\Delta s' = \frac{2.69 h_R^2}{\lambda}, \quad (3.37)$$

where h_R is the Rayleigh criterion resolution of the system. Note that the radius of curvature R in Eq. (3.33) is estimated to be equal the focal length f in Eq. (3.36). The depth of focus is commonly approximated as [Vikram, 1992, p. 62]

$$\Delta s' = \frac{h_R^2}{\lambda}. \quad (3.38)$$

The above development, as mentioned, is based on the Rayleigh criterion resolution limit of a system. However, from the development earlier on the maximum theoretical resolution due to a limited scan area, the actual resolution is worse than the Rayleigh criterion. Therefore, it is more appropriate to use Eq. (3.35) directly for our non diffraction-limited case. If R in Eq. (3.35) is assumed to be roughly equal to the distance d , and assuming $2r$ is equal to the length L of the limited scan area, then the resulting depth of focus is

$$\Delta s' = \pm \frac{2\lambda d^2}{L^2}. \quad (3.39)$$

All of the developments above make the assumption that the aperture is circular. However, since a video camera is being used to digitize the hologram, the aperture is actually rectangular because the electronic imager is inherently rectangular. Thus, a development similar to Born and Wolf cited earlier except based on a rectangular aperture should be applied. Indeed, Born, Wolf, 1980, p. 440 cited Matthews and Cullen for their study of a square microwave lens. Their work for a uniformly illuminated square aperture can be summarized as follows [Matthews, Cullen, 1956, p. 449-453]. The scalar wave function was determined to be [Matthews, Cullen, 1956, p. 452-453]

$$\phi = \frac{4j\eta_o^2}{\lambda d^2} \exp(-jkz') \frac{\pi}{q} \left[C\left(\sqrt{\frac{q}{\pi}}\right) + jS\left(\sqrt{\frac{q}{\pi}}\right) \right]^2, \quad (3.40)$$

where $C(x)$ and $S(x)$ are Fresnel integrals defined by

$$C(x) + jS(x) = \int_0^x \exp\left(\frac{j\pi t^2}{2}\right) dt, \quad (3.41)$$

and

$$q = \frac{kz\eta_o^2}{d^2}. \quad (3.42)$$

The magnitude and phase of Eq. (3.40) applies for q positive, however the magnitude is correct for q positive or negative.

The depth of focus is derived, as in Born and Wolf, by considering the drop in amplitude in the z' direction. The first step is to determine the intensity of the wave at $z' = 0$ from Eq. (3.40). This was determined by substituting Eqs. (3.41) and (3.42) into Eq. (3.40) and calculating $|\phi(0)|^2$ which gave (with the help of Mathcad by Mathsoft)

$$\begin{aligned} |\phi(0)|^2 &= I_o = \lim_{z' \rightarrow 0^+} \left[\left(\frac{4j\eta_o^2}{\lambda d^2} \right) \exp(-jkz') \left(\frac{\lambda d^2}{2z'\eta_o^2} \right) \left[\int_0^{\sqrt{\frac{2z'\eta_o^2}{\lambda d^2}}} \exp\left(\frac{j\pi t^2}{2}\right) dt \right]^2 \right]^2 \\ &= \frac{16\eta_o^4}{\lambda^2 d^4}. \end{aligned} \quad (3.43)$$

Using the same criterion for depth of focus, that is when the intensity has dropped to 0.8 of the intensity at the focal plane, then the edges of the depth of focus will be located where z' satisfies

$$|\phi(z')|^2 = 0.8 \left(\frac{16\eta_o^4}{\lambda^2 d^4} \right) = \frac{12.8\eta_o^4}{\lambda^2 d^4}. \quad (3.44)$$

Unfortunately, a nice closed form of Eq. (3.44) has not been found. Therefore, it is necessary solve Eq. (3.44) through a numerical technique which, for example, finds the root of

$$|\phi(z')|^2 - \frac{12.8\eta_o^4}{\lambda^2 d^4} = 0. \quad (3.45)$$

As an example, let $\lambda = 632.8 \text{ nm}$, $\eta_o = 466.3 \mu\text{m}$, and $d = 73.25 \text{ mm}$. (Note that $2\eta_o$ is the dimension of one side of the aperture.) Substituting these into Eq. (3.45) and solving for z' (with the help of Mathcad) gives $z' = 5.547 \text{ mm}$. Thus, the depth of focus is

$$\Delta s' = \pm 5.547 \text{ mm}. \quad (3.46)$$

In comparison, if the aperture was circular with a diameter of $932.6 \mu\text{m}$ (the same size as one side of the square aperture), then the depth of focus given by Eq. (3.39) is

$$\Delta s' = \pm 7.808 \text{ mm}. \quad (3.47)$$

To determine the relation between depth of focus and depth of field, we start with the thin lens equation, which can be written as

$$\frac{1}{s} + \frac{1}{s'} = \frac{1}{f}, \quad (3.48)$$

where s is the object distance from the lens, s' is the image distance, and f is the focal length of the lens. Equation (3.48) can be rewritten as

$$\frac{1}{s} = \frac{1}{f} - \frac{1}{s'}, \quad (3.49)$$

The derivative of Eq. (3.49) then gives the relation between depth of field and depth of focus as

$$ds = -\frac{s^2}{s'^2} ds' . \quad (3.50)$$

For objects far away from the lens, then s' is approximately equal to the focal length of the lens, and so Eq. (3.50) can be written as approximately

$$ds = -\frac{s^2}{f^2} ds' . \quad (3.51)$$

To determine the depth of field in holograms, it is helpful to depict the reconstruction of the real image from a hologram as shown in Fig. 3.18 [Born, Wolf, 1980, p. 454], [Kreis, 1996, p. 157]. In the case of digital reconstruction, the lens can be numerically simulated and positioned exactly at the hologram plane [Kreis, 1996, p. 157]. If no magnification is desired, then $s = s'$ and so the magnitudes of depth of field and depth of focus are equal.

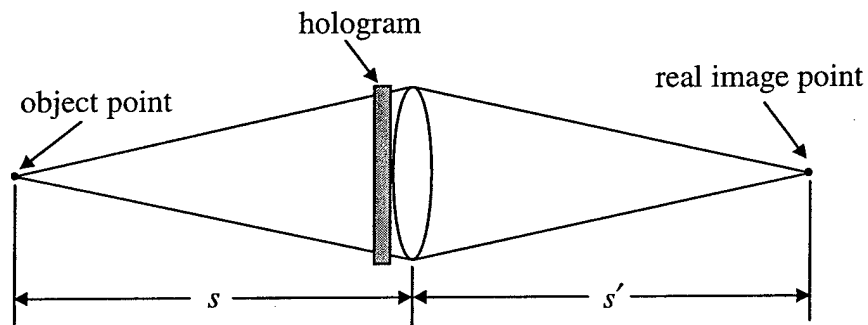


Figure 3.18 Real image reconstruction geometry.

4.0 EXPERIMENTAL DESIGN

4.1 Objective of Experiments

There are two main objectives to be accomplished through experimentation. The first is to verify the resolution achievable for a limited digitized area of a hologram, as discussed in Chapter 3. The approach will be to make a hologram of a resolution target. Such targets have lines or bars of known spatial resolution. The second goal is to demonstrate the depth of field expected in digital reconstructions. The approach will be to make a hologram of a custom designed "ruler" that is oriented at an angle with respect to the hologram plane. In this way, certain parts of the ruler depending on the reconstruction distance.

4.2 Components

Before the experiments can be designed, the hardware available for use must be described since their specifications will be a factor in the actual designs. The object used to measure resolution is a standard United States Air Force 1951 target. A close-up view of the target is provided in Fig. 4.1. The chart has sets of three lines, both horizontal and vertical, that range in spatial frequency from 1.00 to 228.0 line pairs per millimeter (lp/mm). The camera used for digitizing the holograms is a Kodak Megaplug 1.4 which uses a CCD imager having 1316 x 1033 pixels where each pixel is 6.8 μm square and the fill ratio is 100 percent. A Matrox frame grabber was used for obtaining and storing digitized images from the camera. A standard Pentium-based personal computer was used communicate with the frame grabber and store the digitized images onto disk. A Nikon microscope with a 10X objective having a numerical aperture of 0.25 was used. The magnification M was determined experimentally to be 7.4664. How this was determined will be discussed in the next chapter. A Hughes model 3224H-PC helium-neon laser was used to make the holograms and so the wavelength, λ , is 632.8nm.

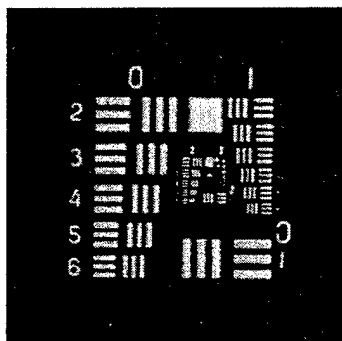


Figure 4.1 United States Air Force 1951 resolution target.

4.3 Resolution Test

The objective of this experiment is to demonstrate that the resolution limitation due to a limited digitized area of a hologram is as described by Eq. (3.29). The approach to verify this equation through experimentation is illustrated in Fig. 4.2. There are several parameters of the experimental setup that must be taken into consideration to successfully demonstrate the resolution achievable. These parameters are the distance of the resolution chart from the film plate, the frequency of the resolution bars on the chart, the orientation and wavelength of the object and plane waves, the numerical aperture and magnification of the microscope objective, and the wavelength of the microscope illumination. Explanation of each of these parameters now follows.

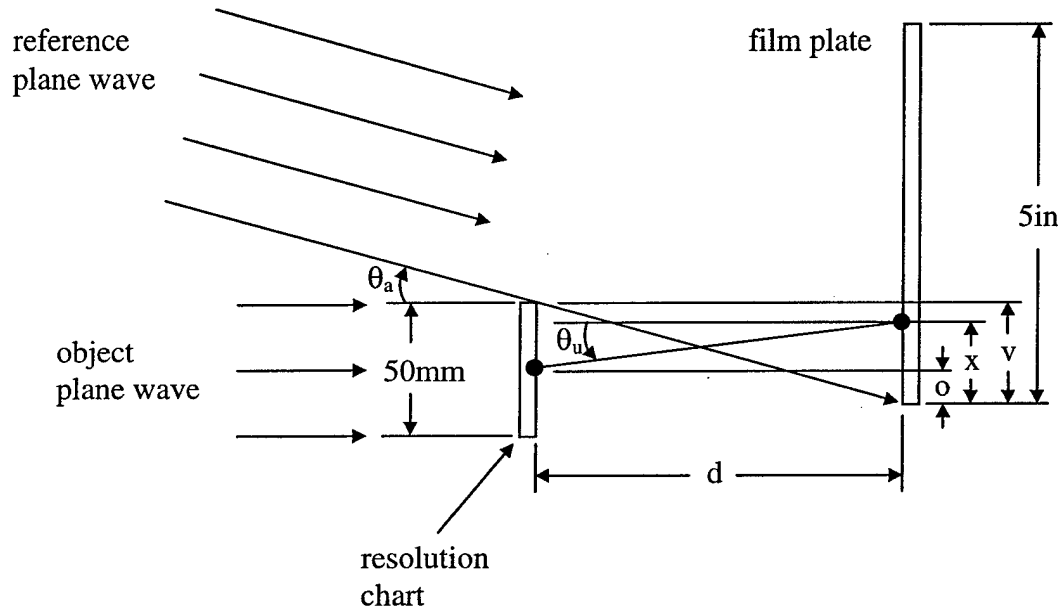


Figure 4.2 Resolution test setup.

The first parameter considered is the frequency of the resolution bars on the chart. The sets of lines that can be resolved in this experiment are determined by Eq. (3.29). Since a microscope is used to digitize the hologram, the magnification must be included in calculating L in Eq. (3.29). Defining C as the actual size of the imager, then

$$L = \frac{C}{M}. \quad (4.1)$$

The number of pixels in the imager multiplied by their size gives C . Substituting Eq. (4.1) into Eq. (3.29) gives

$$\alpha_o = \frac{C}{2M\lambda d}, \quad (4.2)$$

where the subscript o indicates the frequency is for the object (as opposed to the hologram which will be addressed later).

Substituting all of the parameters given earlier in Eq. (4.2) and taking into consideration the practicality of the distance the target could be placed from the film plate, it was decided to place the target at distances of 164mm, 218mm, and 350mm. Actually, the hologram of the target at distances of 164mm and 350mm was done as a double exposure for the purpose of the depth of field experiment. The expected resolution of the target achievable for the distances of 164mm, 218mm, and 350mm were 4.493 lp/mm, 3.380 lp/mm, and 2.105 lp/mm, respectively. On the resolution target, these spatial frequencies correspond to Group 2 Element 2, Group 1 Element 5, and Group 1 Element 1, respectively.

For this same experiment, there is another spatial frequency resolution problem that must be addressed. That is the spatial frequency of the fringes created on the hologram due to the angles of the object and reference waves. The spatial frequency resulting from the interference of the object and reference waves was derived in Chapter 2 and the result [Eq. (2.34)] was

$$\alpha_h = \frac{\sin(\theta_a) + \sin(\theta_u)}{\lambda}, \quad (4.3)$$

where the subscript h is used to signify the hologram. From Fig. 4.1, the geometry of the setup can be used to derive the angles θ_a and θ_u as

$$\theta_a = \tan^{-1}\left(\frac{v}{d}\right) \quad (4.4)$$

and

$$\theta_u = \tan^{-1}\left(\frac{x-o}{d}\right). \quad (4.5)$$

Substituting Eqs. (4.4) and (4.5) into Eq. (4.3) gives

$$\alpha_h = \frac{\sin\left(\tan^{-1}\left(\frac{v}{d}\right)\right) + \sin\left(\tan^{-1}\left(\frac{x-o}{d}\right)\right)}{\lambda}. \quad (4.6)$$

The importance of Eq. (4.6) is that a microscope objective has a limit to its resolution capability usually expressed as the numerical aperture (NA) as discussed in Chapter 3. Therefore, the geometry of the experimental setup must be adjusted so that the microscope can resolve the interference fringes recorded on the hologram. The NA required to resolve the spatial frequencies described by Eq. (4.6) can be expressed by combining Eqs. (3.7) and (3.29) which gives

$$NA = 0.61 \cdot 2 \cdot \lambda_m \cdot \alpha_h, \quad (4.7)$$

where the subscript m is used to indicate that λ_m is the wavelength of the light used by the microscope to illuminate the hologram. The microscope contains a green interference filter with a specified center frequency of 546nm. The filter bandwidth around this center frequency is not known. The longest wavelength in the bandwidth would correspond to

the most stringent NA requirement. The longest wavelength is estimated to be 600nm, and so that was the wavelength used for λ_m in Eq. (4.7).

Table 4.1 provides a number of NAs for various object and hologram locations to show the variations that exist. The first four parameters in Table 4.1, namely d , v , x , and o , are shown in Fig 4.2. The parameter d is the distance the resolution target is from the hologram film plate. The parameter v is the distance the edge of the resolution target is from the edge of the hologram film plate. The parameters x and o are specific locations on the hologram plate and resolution target, respectively. As can be seen, the NA can vary significantly, depending on what part of the hologram is being digitized. Because of these variations, it is important to consider the geometry of the setup when deciding what part of the hologram to digitize.

Table 4.1 Effect of various parameters on the numerical aperture.

d (mm)	v (mm)	x (mm)	o (mm)	λ (nm)	α_b (lp/mm)	λ_m (nm)	NA
164.00	34.00	0.00	0.00	632.80	320.80	600.00	0.23
164.00	34.00	0.00	15.00	632.80	176.86	600.00	0.13
164.00	34.00	0.00	34.00	632.80	0.00	600.00	0.00
164.00	34.00	15.00	0.00	632.80	464.73	600.00	0.34
164.00	34.00	15.00	15.00	632.80	320.80	600.00	0.23
164.00	34.00	15.00	34.00	632.80	138.93	600.00	0.10
164.00	34.00	34.00	0.00	632.80	641.59	600.00	0.47
164.00	34.00	34.00	15.00	632.80	502.66	600.00	0.37
164.00	34.00	34.00	34.00	632.80	320.80	600.00	0.23

Figure 4.3 is a picture of the actual setup used in the experiment. In order to facilitate the positioning of the target, hologram film plate, object wave, and plane wave, the edges of the target and hologram film plate were used as guides. This can be seen by referring back to Fig. 4.2. In Fig. 4.2, the long vector of the reference plane wave illustrates how these edges were used. The reference plane wave was positioned so that it just barely missed the edge of the resolution target and landed on the edge of the hologram as shown. This made positioning the plane wave at the desired angle easier.

The entire resolution target was not placed directly in front of the film plate. The target and film holder were adjusted so that the portion of the target with the lines just appeared at the edge of the film plate. This allowed using a smaller plane wave angle to reduce the spatial frequencies on the hologram.

4.4 Depth of Field Test

In order to test the depth of field, two different approaches were taken. First, a double exposure hologram was made with the resolution target placed at two different

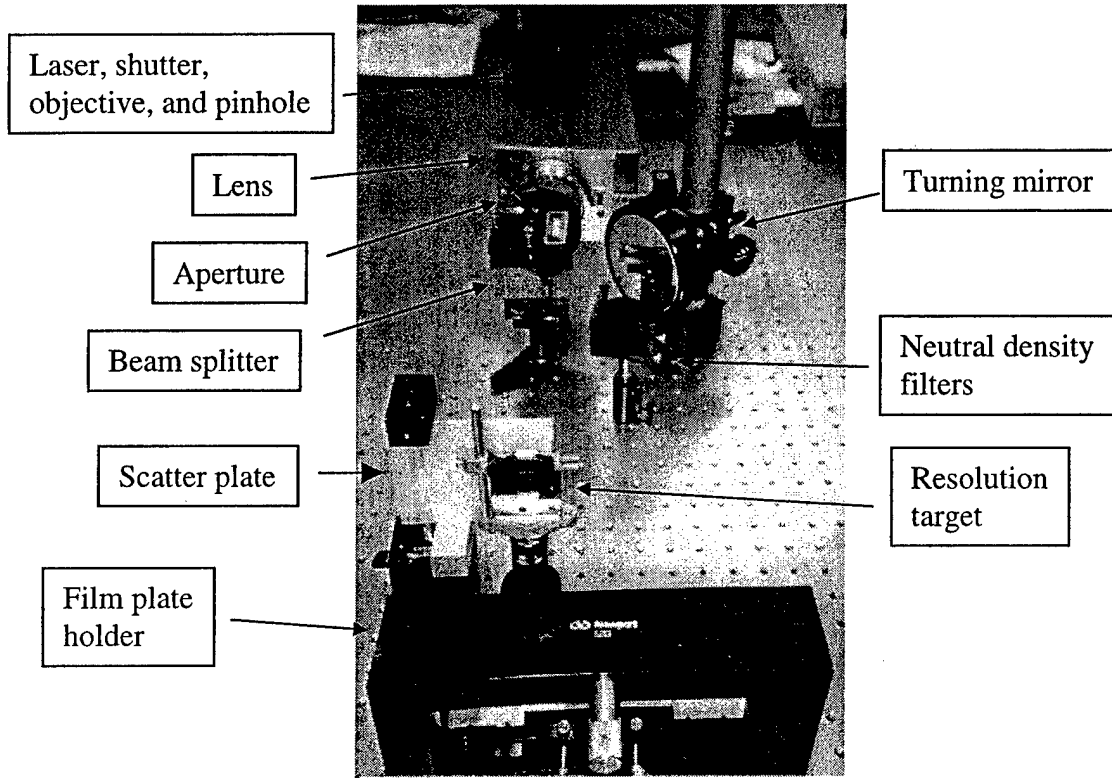


Figure 4.3 Resolution test setup.

distances from the hologram film plate. The second approach was to make a hologram of a ruler at a slanted angle. The depth of field can be calculated by solving for the z'

which satisfies Eq. (3.44) which when combined with Eq. (4.1) and noting that $\eta_o = \frac{L}{2}$ becomes

$$|\phi(z')|^2 - \frac{0.8C^4}{\lambda^2 M^4 d^4} = 0. \quad (4.8)$$

The depth of field is then

$$\Delta s = \pm z'. \quad (4.9)$$

For the first approach, the setup is the same as in Figs. 4.2 and 4.3 except that the resolution target was placed at two different locations. Since depth of field is more of a subjective parameter, it is difficult to measure. So, the two distances chosen were such that the degradation would definitely be noticeable. Since, a hologram with distance of 164mm from the hologram film plate had already been chosen for the resolution test, it was decided to use this as one of the distances. At this distance, the total depth of field from Eqs. (4.8) and (4.9) is

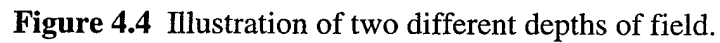
$$\Delta s = \pm 27.8mm. \quad (4.10)$$

This indicated that the second distance would have to be at least

$$164mm + 27.8mm = 191.8mm. \quad (4.11)$$

$$\Delta s = \pm 126.6 \text{ mm} . \quad (4.12)$$

Figure 4.4 shows the relation between the two depths of field and distances.



43

the hologram film plate. This ruled out use of traditional rulers. So, a ruler was designed and printed on a transparency using a laser printer. The ruler used is shown in Fig. 4.6. Note that three of the lines are only half the length of the others. This was done to help in identifying what part of the ruler has been reconstructed.

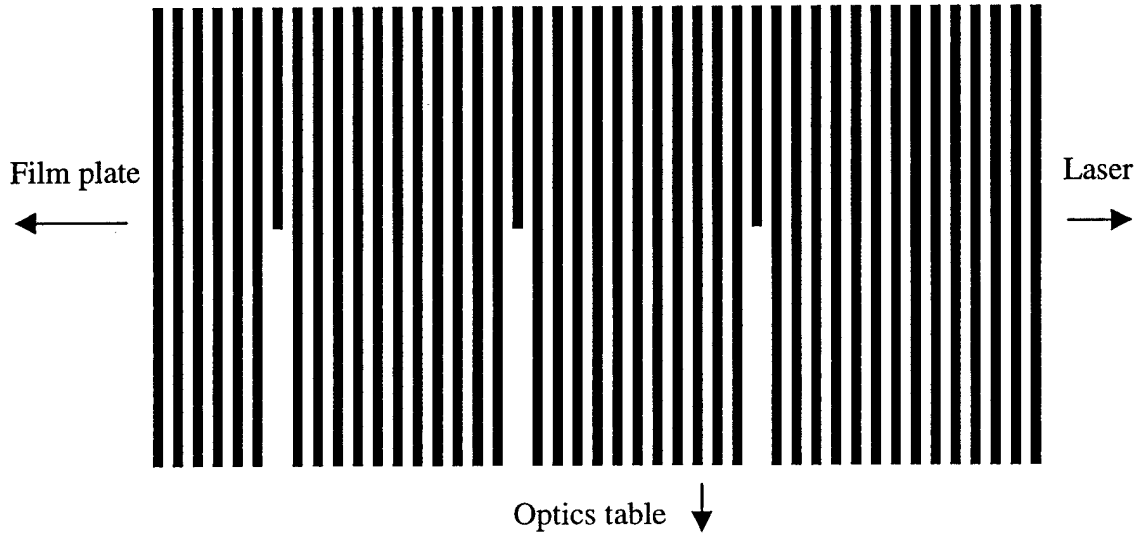


Figure 4.6 Line pattern used as a ruler in one depth of field test.

For this experiment, there were a number of parameters to consider in the design process. Besides the NA, it was also necessary to consider the angle of the ruler and how that would affect the apparent size of the individual ruler lines. The angle was somewhat controlled by the width of the object plane wave. The NA affected the width and the length of the ruler. After consideration of all of these factors, it was decided that the distance would be set to 185mm, the width would be set to 24mm, and the length of the ruler would be set to 114.3mm.

Determining the NA for this setup is not as straightforward as in the resolution target setup. This is because the object (ruler) varies in distance from the hologram film plate. From Fig. 4.5, the angle of the reference wave is given by

$$\theta_a = a \tan\left(\frac{v}{d}\right). \quad (4.13)$$

The angle of the object wave is given by

$$\theta_u = a \tan\left(\frac{x-o}{d - \frac{(v-o)}{\tan(\theta_r)}}\right). \quad (4.14)$$

Table 4.2 shows how the NA varies for this geometry. Figure 4.7 is a picture of the setup used for this depth of field test. Similar to the resolution test, the reference plane wave was adjusted so that it just barely missed the edge of the ruler farthest from the film plate.

Table 4.2 Numerical aperture values for depth of field test setup.

d (mm)	v (mm)	x (mm)	o (mm)	λ (nm)	α_h (lp/mm)	λ_m (nm)	NA
185.00	24.00	0.00	0.00	632.80	203.31	600.00	0.15
185.00	24.00	0.00	15.00	632.80	38.55	600.00	0.03
185.00	24.00	0.00	24.00	632.80	0.00	600.00	0.00
185.00	24.00	15.00	0.00	632.80	520.34	600.00	0.38
185.00	24.00	15.00	15.00	632.80	203.31	600.00	0.15
185.00	24.00	15.00	24.00	632.80	126.52	600.00	0.09
185.00	24.00	24.00	0.00	632.80	695.34	600.00	0.51
185.00	24.00	24.00	15.00	632.80	302.50	600.00	0.22
185.00	24.00	24.00	24.00	632.80	203.31	600.00	0.15

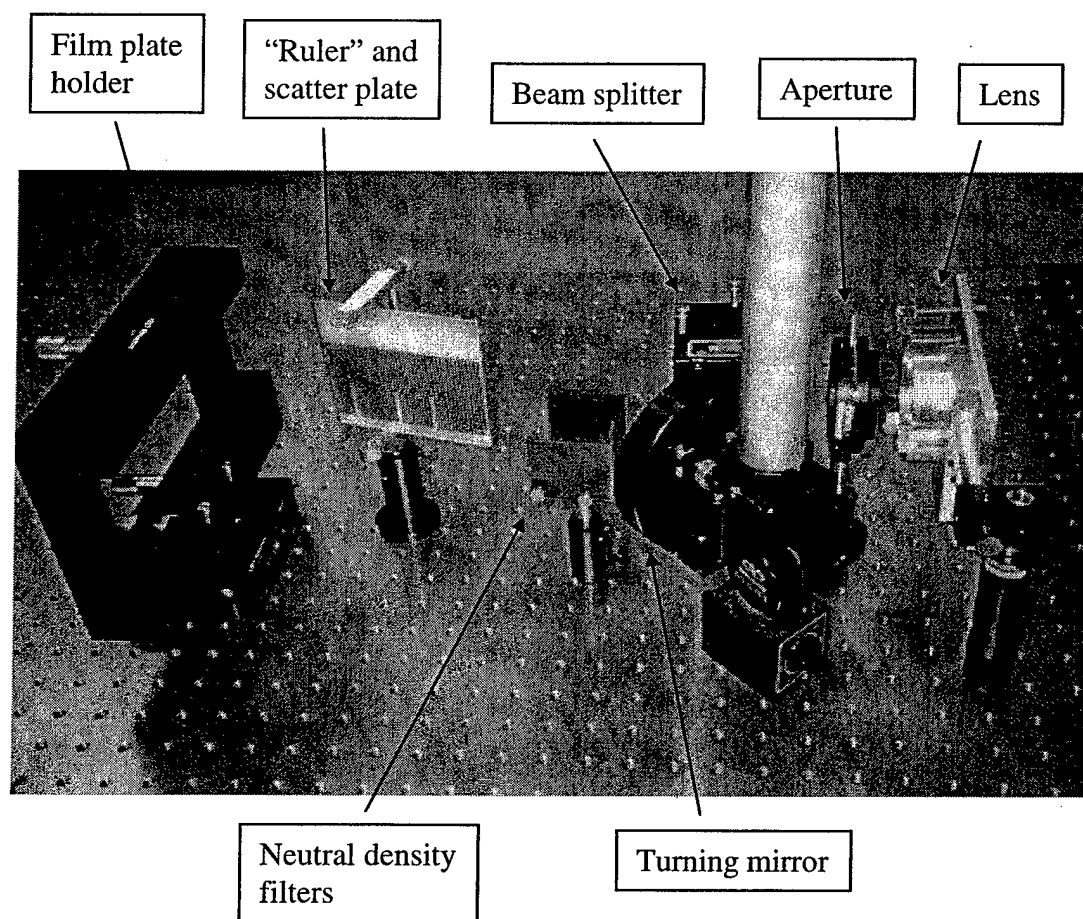


Figure 4.7 Depth of field test setup.

5.0 DIGITAL HOLOGRAPHY RECONSTRUCTION PROGRAM

In this chapter, the mathematics and software code used in the digital reconstruction process will be explained. The software used in the development of this program is called IDL by Research Systems, Inc. In this section only the essential math in the process will be addressed. The full program including all of the data I/O, GUIs, etc is included in Appendix C. In Chapter 2, the equation for calculating the wavefront at the hologram plane was given as

$$\mathbf{U}(\xi, \eta) = \frac{e^{jkd}}{j\lambda d} \exp\left[\frac{jk}{2d}(\xi^2 + \eta^2)\right] \iint_{\Sigma} \mathbf{O}(x, y) \exp\left[\frac{jk}{2d}(x^2 + y^2)\right] \exp\left[-\frac{j2\pi}{\lambda d}(x\xi + y\eta)\right] dx dy. \quad (5.1)$$

In the reconstruction process, Eq. (5.1) can also be used except that now the equation becomes

$$\mathbf{O}(x', y') = \frac{e^{jkd}}{j\lambda d} \exp\left[\frac{jk}{2d}(x'^2 + y'^2)\right] \iint_{\Sigma} \mathbf{U}(\xi, \eta) \exp\left[\frac{jk}{2d}(\xi^2 + \eta^2)\right] \exp\left[-\frac{j2\pi}{\lambda d}(x'\xi + y'\eta)\right] d\xi d\eta, \quad (5.2)$$

where $\mathbf{U}(\xi, \eta)$ is the wavefront at the hologram plane and $\mathbf{O}(x', y')$ is the wavefront at the reconstruction plane. In order to obtain $\mathbf{U}(\xi, \eta)$ for use in the calculations, the digitized portion of the hologram must be multiplied by the reference plane wave. The result is $\mathbf{U}(\xi, \eta)$. Next in the equation is

$$\exp\left[\frac{jk}{2d}(\xi^2 + \eta^2)\right]. \quad (5.3)$$

Equation (5.3) is sometimes referred to as a Gaussian phase term. Multiplying Eq. (5.3) with $\mathbf{U}(\xi, \eta)$ and then taking the forward Fourier transform gives the double integral portion of Eq. (5.1). Multiplying this by the terms in front of the double integrals in Eq.(5.1) gives the final desired result $\mathbf{O}(x', y')$.

Mathematically creating a reference plane wave begins with the basic equation for describing electromagnetic (EM) wave propagation. Defining the reference plane wave as $\mathbf{P}(\xi, \eta)$, the general equation is

$$\mathbf{P}(\xi, \eta) = \exp j(\vec{k} \cdot \vec{r} - \omega t + \phi). \quad (5.4)$$

The last two factors in the exponent can be ignored. The dot product is given by expressing \vec{k} in its ξ and η components. Using the spherical coordinate system, the plane wave can be described by

$$\mathbf{P}(\xi, \eta) = \exp j[k \cos(\theta) \sin(\phi) + k \sin(\theta) \sin(\phi)]. \quad (5.5)$$

Defining

$$k_{\xi} = k \cos(\theta) \sin(\phi) \quad (5.6)$$

and

$$k_{\eta} = k \sin(\theta) \sin(\phi), \quad (5.7)$$

Eq. (5.5) becomes

$$P(\xi, \eta) = \exp j(k_{\xi} + k_{\eta}). \quad (5.8)$$

The software code that creates the plane wave is the following (note that throughout the software code the letters ξ and η were replaced with x and y because of the difficulty in using Greek letters in the code):

```
k=2.*!dpi/info.lambda
kx=k*cos(info.theta*!dpi/180d)*sin(info.phi*!dpi/180d)
ky=k*sin(info.theta*!dpi/180d)*sin(info.phi*!dpi/180d)
vectx=(dindgen(info.mc)-round(info.mc/2))*info.dx/info.mag
arx=transpose(make_array(info.nc,1,/double,value=1.0))
vecty=(transpose(dindgen(info.nc)-round(info.nc/2))*info.dy/info.mag
ary=make_array(info.mc,1,/double,value=1.0)
planewave=kx*(vectx#arx)+ky*(ary#vecty)
planewave=exp(dcomplex(0,temporary(planewave)))
```

In the above code, the variable *info* is a named structure that is basically a container of variables. So, *info.lambda* refers to the variable *lambda* contained in the structure *info*. The variables in *info* called in the above code are *lambda* which is the laser wavelength, *theta* and *phi* which define the orientation of the plane wave in spherical coordinates, *mc* and *nc* which are the dimensions of the digitized hologram, *dx* and *dy* which are the actual size of the pixels in the camera, and finally *mag* which is the actual magnification of the microscope. In addition, *!dpi* calls the double precision value of π . Whenever *d* is used next to a number as in *180d*, this defines the number as double precision floating point number. The result of the code is that it calculates the plane wave as illustrated in the equation where the digitized hologram size is assumed to be 1024 X 1024 pixels:

$$\begin{aligned} \text{planewave} = \exp j \left\{ \begin{bmatrix} -512 & -511 & \cdots & 511 \\ -512 & -511 & \cdots & 511 \\ \vdots & \vdots & \vdots & \vdots \\ -512 & -511 & \cdots & 511 \end{bmatrix} \cdot \frac{kx \cdot dx}{mag} \right. \\ \left. + \begin{bmatrix} 511 & 511 & \cdots & 511 \\ 510 & 510 & \cdots & 510 \\ \vdots & \vdots & \vdots & \vdots \\ -512 & -512 & \cdots & -512 \end{bmatrix} \cdot \frac{ky \cdot dy}{mag} \right\}. \end{aligned} \quad (5.9)$$

Multiplying the result of Eq. (5.9) with the digitized portion of the hologram creates $U(\xi, \eta)$ in Eq. (5.2).

The next term in Eq. (5.2) after $U(\xi, \eta)$ is the Gaussian phase term given in Eq. (5.3). If the original plane wave and not the conjugate plane wave is used to recreate the object wavefront at the hologram plane, then a Fourier lens must be introduced to focus

the real image at the observation plane. The phase transformation of a lens with a focal length of f is [Goodman, 1996, p. 99]

$$t(\xi, \eta) = \exp\left[-j \frac{k}{2f} (\xi^2 + \eta^2)\right]. \quad (5.10)$$

Taking the approach of multiplying the wavefront at the hologram plane by the transformation of Eq. (5.10) results in essentially placing a lens of focal length f at a distance d from both the object and image. Thus, the focal length of the lens is

$$f = \frac{d}{2}. \quad (5.11)$$

Mathematically, the lens transformation Eq. (5.10) is accounted for by placing it into Eq. (5.2) resulting in [Goodman, 1996, p. 102-104]

$$\begin{aligned} \mathbf{O}(x', y') &= \frac{e^{jkd}}{j\lambda d} \exp\left[\frac{jk}{2d}(x'^2 + y'^2)\right] \\ &\cdot \iint_{\Sigma} \mathbf{U}(\xi, \eta) t(\xi, \eta) \exp\left[\frac{jk}{2d}(\xi^2 + \eta^2)\right] \exp\left[-\frac{j2\pi}{\lambda d}(x'\xi + y'\eta)\right] d\xi d\eta. \end{aligned} \quad (5.12)$$

Substituting Eqs. (5.10) and (5.11) into Eq. (5.12) gives

$$\begin{aligned} \mathbf{O}(x', y') &= \frac{e^{jkd}}{j\lambda d} \exp\left[\frac{jk}{2d}(x'^2 + y'^2)\right] \\ &\cdot \iint_{\Sigma} \mathbf{U}(\xi, \eta) \exp\left[\frac{-jk}{2d}(\xi^2 + \eta^2)\right] \exp\left[-\frac{j2\pi}{\lambda d}(x'\xi + y'\eta)\right] d\xi d\eta. \end{aligned} \quad (5.13)$$

Note that the difference between Eqs. (5.1) and (5.13) is the Gaussian phase term of Eq. (5.1) is replaced by its conjugate to give Eq. (5.13).

The Gaussian phase term of Eq. (5.13) is created by the following software code:

```
gauss=-((vectx#arx)^2+(ary#vecty)^2)*k/2./info.d
gauss=exp(complex(0,temporary(gauss))).
```

The variable *info.d* refers to the distance of the object from the hologram (and also the distance from the hologram to the real image). The Gaussian phase term as calculated by the software code can be illustrated as follows where again the digitized hologram size is assumed to be 1024 X 1024 pixels:

$$\begin{aligned} gauss &= \exp\left(\frac{-jk}{2d}\right) \\ &\cdot \left\{ \left[\begin{bmatrix} -512 & -511 & \dots & 511 \\ -512 & -511 & \dots & 511 \\ \vdots & & & \\ -512 & -511 & \dots & 511 \end{bmatrix} \cdot \frac{dx}{mag} \right]^2 + \left[\begin{bmatrix} 511 & 511 & \dots & 511 \\ 510 & 510 & \dots & 510 \\ \vdots & \vdots & & \vdots \\ -512 & -512 & \dots & -512 \end{bmatrix} \cdot \frac{dy}{mag} \right]^2 \right\}. \end{aligned} \quad (5.14)$$

The final steps in reconstructing the image from the hologram are fairly straightforward. The software code lines are as follows:

```
recon=double(holo)*gauss*planewave
recon=fft(temporary(recon),-1,double=1,/overwrite)
recon=abs(temporary(recon))^2
recon=shift(temporary(recon),info.mc/2,info.nc/2).
```

The first step is to multiply the digitized portion of the hologram (called *holo*), the plane wave calculated in Eq. (5.9), and the Gaussian phase term in Eq. (5.14) together. The result is called *recon*. Then a forward Fourier transform of *recon* is calculated and placed back into *recon* (to save memory). The absolute value squared of *recon* is calculated since the desired result is the intensity because that is what we see with our eyes. Finally, the image is shifted because the discrete Fourier transform gives the positive frequencies first followed by the mirror image or negative frequencies.

The Graphical User Interface (GUI) is shown in Fig. 5.1. The general process is to load the interference image to be reconstructed and then input the various parameters such as the distance of the object from the hologram film plate, magnification of the

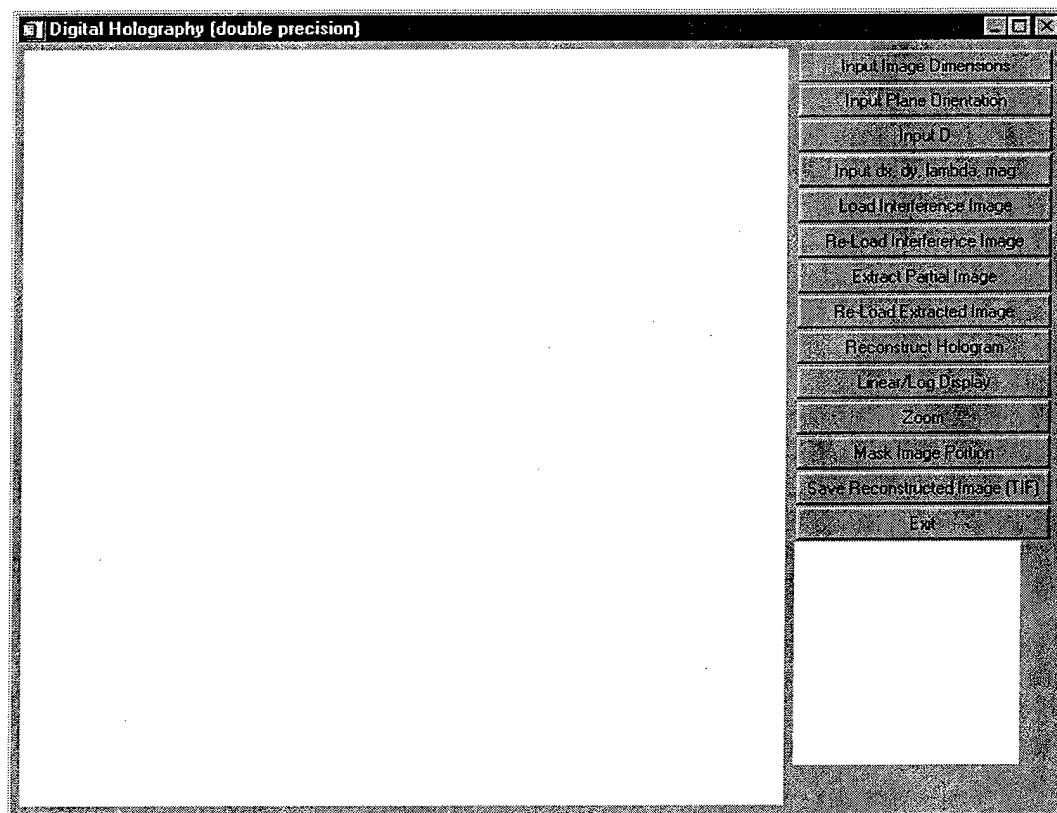


Figure 5.1 Graphical User Interface (GUI) of the reconstruction program.

microscope, size of the pixels, etc.. The buttons for partial images allow cutting the size of an image so that, for instance, the size can be adjusted to powers of 2. The Linear/Log Display button is used to switch between the raw reconstructed image being displayed or to take the log of the raw reconstructed image and then displaying it. The latter helps to bring out detail in reconstructions that have a large dynamic range. The Mask Image Portion provides an additional method of bringing out detail in reconstructed images. It allows brighter parts of the image to be suppressed by essentially drawing a black box on top of them.

6.0 EXPERIMENTAL RESULTS

6.1 Microscope Magnification

As discussed in Chapter 4, the magnification of the microscope objective must be included in the calculations because it directly affects the size of the hologram digitized by the camera. Therefore, the actual amount of magnification must be determined. The method of determining the magnification was to digitize the resolution target itself with the microscope and camera. Since the sizes of the lines on the resolution target are known, then the magnification can be determined. In order to reduce the error as much as possible, the largest resolution target line that would fit into the field of view was chosen. This was determined to be Group 0 Element 1, this largest on the chart. Figure 6.1 shows the digitized image. The brighter pixels are the line, which fills about half of the vertical field of view.

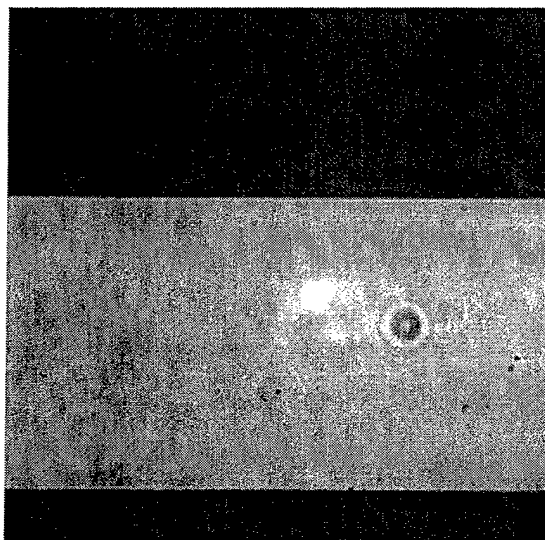


Figure 6.1 Digitized image of a Group 0 Element 1 line from the resolution target.

From the specifications of the resolution target, it is known that Group 0 Element 1 lines are 1.00 line pairs/mm. Therefore, one line is 0.5mm in width. If no magnification were used, then one line would correspond to

$$\left(\frac{0.5\text{mm}}{\text{line}} \right) \left(\frac{1\text{pixel}}{6.8\mu\text{m}} \right) = 73.53 \frac{\text{pixels}}{\text{line}}. \quad (6.1)$$

In other words, the resolution target line would subtend 73.53 pixels on the camera's imager. Now, if the number of pixels subtended by the same line under the microscope can be determined, then the magnification can be calculated. Using MATLAB to read and zoom in on Fig. 6.1, the number of pixels can be determined. Figures 6.2 and 6.3 show the upper and lower edges, respectively. At this time, the reader is referred to

Appendix D which discusses relative error propagation. It is necessary in understanding how the errors discussed in the remainder of this chapter are estimated and calculated.

Figures 6.2 and 6.3 illustrate the difficulty in determining the exact magnification. Deciding the location of the edges precisely is not possible. This is where the error analysis and propagation discussed earlier must be applied. From Fig. 6.2, it appears that the upper edge is located at pixel number 368. However, this may not be the true edge. A reasonable estimate of the error is ± 2 pixels. From Fig. 6.3, it appears that the lower edge is located at pixel number 917. Again, the estimate of error is ± 2 pixels. The resulting width and corresponding relative error of the line is then

$$549 \text{ pixels} \pm 4 \text{ pixels} = 550 \text{ pixels} \pm 0.7286\% . \quad (6.2)$$

(Keep in mind that the relative error is actually approximated since the measured as opposed to the true value is used as discussed earlier in this chapter.) The actual magnification of the microscope objective can now be calculated to be

$$M = \frac{549 \text{ pixels} \pm 0.7286\%}{73.53 \text{ pixels}} = 7.4664 \pm 0.7286\% . \quad (6.3)$$

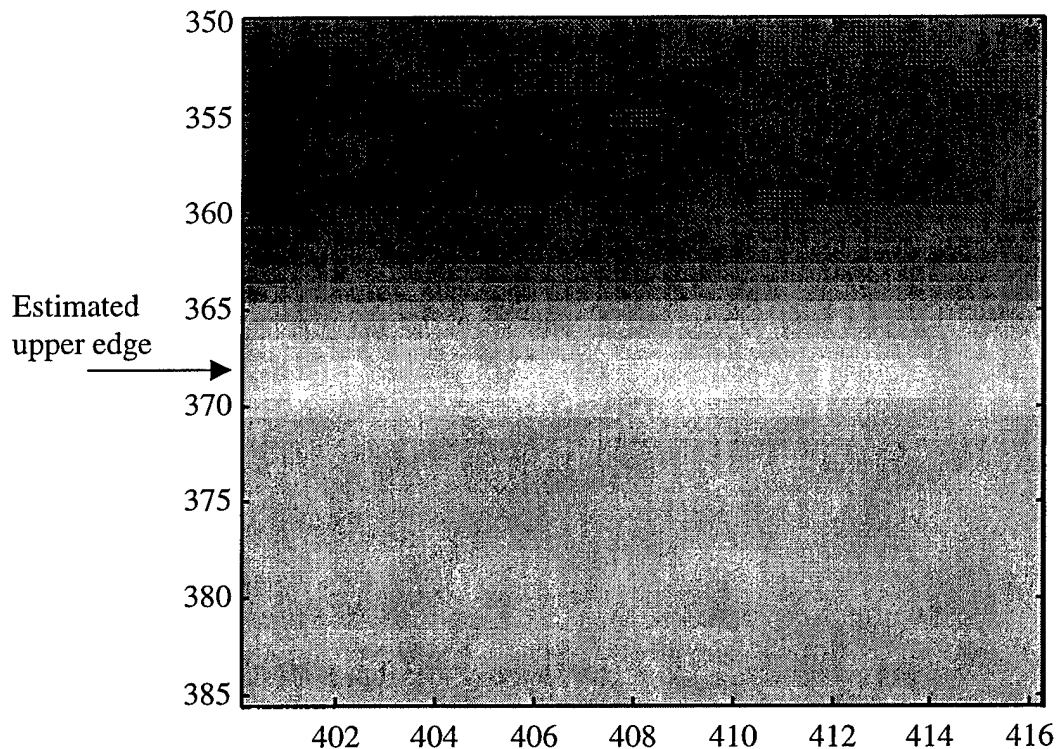


Figure 6.2 Magnified upper edge of resolution line.

6.2 Unfeasibility of Testing Mosaicking

The significant result from section 3.2.2 on mosaicking is that accuracy of placement must be better than one pixel to achieve an improvement in resolution. One

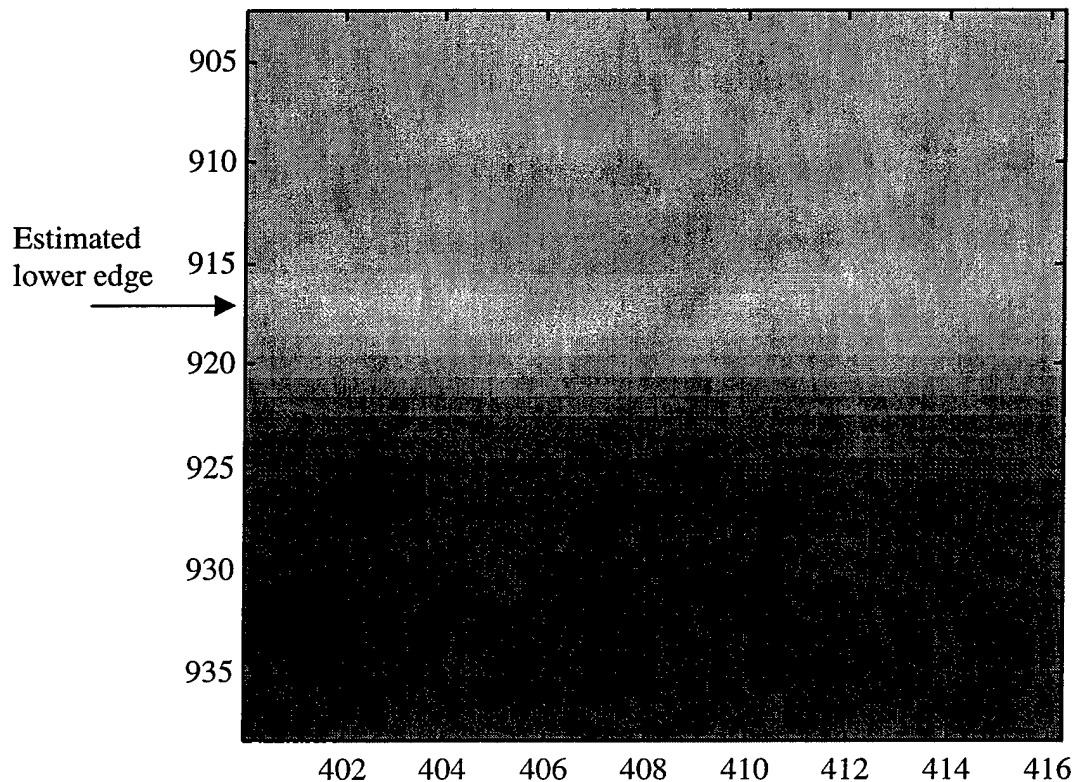


Figure 6.3 Magnified lower edge of resolution line.

pixel of the camera is $6.8\mu\text{m}$. However, when magnification is taken into consideration, one pixel is

$$\frac{6.8\mu\text{m}}{7.4664 \pm 0.7286\%} = 0.9107\mu\text{m} \pm 0.7286\% \quad (6.4)$$

or

$$0.9107\mu\text{m} \pm 0.006636\mu\text{m}. \quad (6.5)$$

Therefore, accuracy of placement must be within $0.9107\mu\text{m}$. The most accurate translation stage available for use in these experiments has a smallest increment of $0.5\mu\text{m}$. Straightness of travel and orientation of the camera with respect to the direction of travel of the stage must also be considered. When all the factors are combined, it was considered that a test of mosaicking was impractical.

6.3 Film Exposure and Developing

The hologram film plates used were 8E75HD-1 Holotest plates manufactured by Agfa. The specified exposure was $\frac{15\mu\text{J}}{\text{cm}^2}$ at the HeNe laser wavelength to achieve an exposure density of 1. The power meter used is a Spectra-Physics model 404. It was not necessary to use an attenuator. The entrance aperture of the detector head is specified as 7mm in diameter, thus the area is 0.385cm^2 . In order to increase visibility of the fringes,

the weak beam from the splitter was used as the reference plane wave and the strong beam was used as the object wave. Since it was necessary to use a scatter plate, the object beam reaching the film plate was still weaker than the reference beam. So, neutral density filters were also used. It was determined experimentally that the two neutral density filters together had a transmission of 0.067.

For the double exposure hologram, the reference beam was measured to be 0.006mW and the object beam (including the scatter plate) was measured to be 0.5uW. These measurements were with the resolution target at a distance of 164mm. When the neutral density filters were included, the reference beam then became 0.402uW. So, the total power at the film plate was approximately 1.0uW. The exposure time can then be calculated as

$$\frac{\frac{15uJ}{cm^2}}{\frac{1uW}{0.385cm^2}} = 5.775s. \quad (6.6)$$

Since this was a double exposure hologram, it was decided to reduce the exposure by a factor of $\sqrt{2}$ and so the exposure then became 4.08s. The actual used was 4.1s because that was the closest setting possible on the shutter controller. For the second exposure with the resolution target at 350mm, a similar procedure led to an exposure of 8.2s. Similarly, for the depth of field test with a ruler as the object the exposure was 8.6s.

The developing of the film plates was done as follows. The developer used was Kodak Dektol developer cat # 146 4726 and the fixer used was Kodak fixer cat # 197 1746. The film plates were placed in the developer for 3min, washed in running water for 1min, placed in the fixer for 5min, washed in running water for 10min, and finally dried.

6.4 Resolution Determination

The resolution limit imposed by the size of the hologram digitized is now addressed. Three different reconstructions are performed and analyzed. The first is a reconstruction from a hologram made of the resolution target at a distance of 218mm from the hologram film plate. The entire reconstruction is given in Fig. 6.4 and Fig. 6.5 is a magnified view of just the resolution target itself. In Fig. 6.4, the real and virtual images have been labeled. Note that these are as predicted in Fig. 2.4. The image in Fig. 6.5 has been rotated so that the resolution target matches the orientation of the one in Fig. 4.1. The reconstruction is noisy, making it difficult to determine the actual resolution. From a visual inspection of Fig. 6.5, it appears that the maximum resolution is about Group 1 Element 4. The expected resolution was $3.38 \frac{lp}{mm}$ which corresponds to about Group 1 Element 5.

The achievable resolution can be estimated by determining the size of the individual pixels that make up the reconstructed image. Since the size of the lines and blocks in the resolution target are known, then the pixel size can be calculated. The best

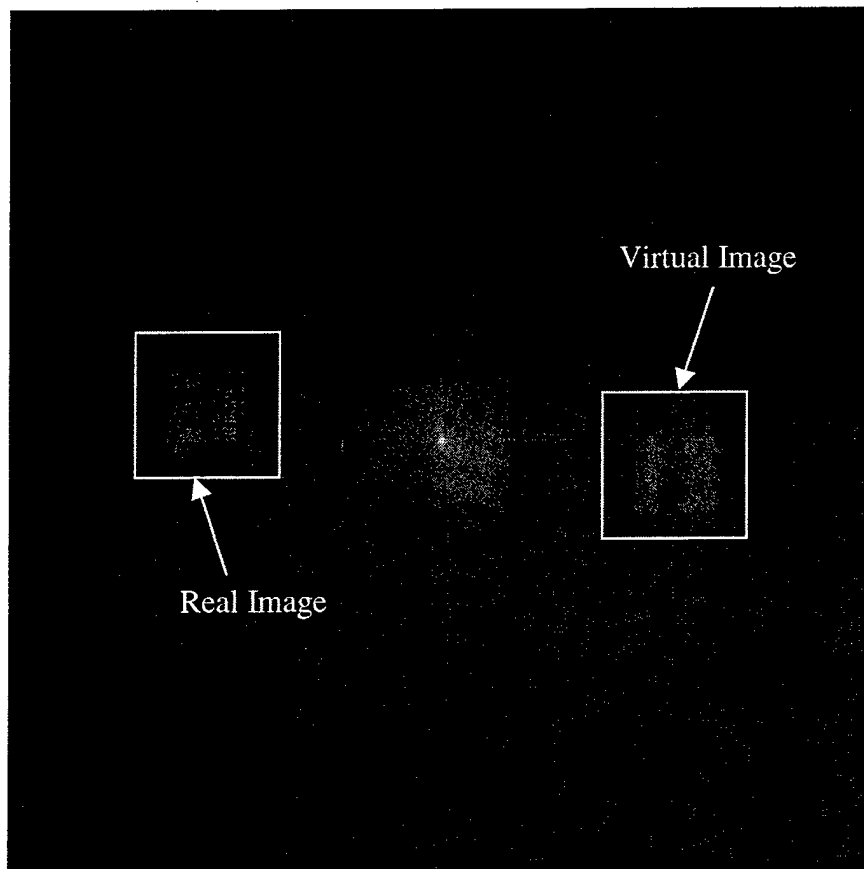


Figure 6.4 Reconstruction of the resolution target at a distance of 218mm.

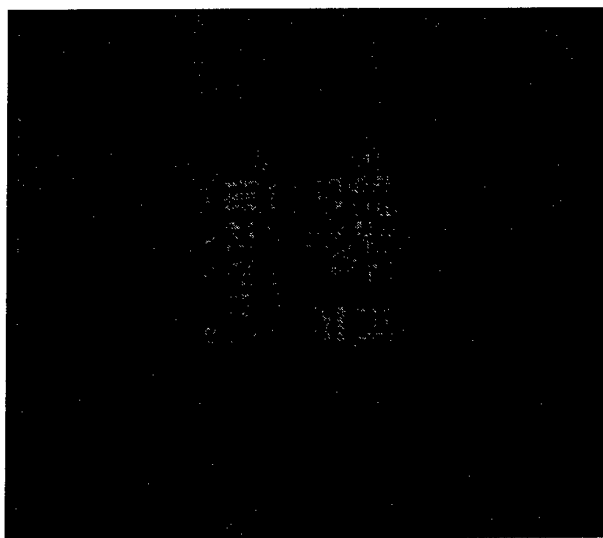


Figure 6.5 Magnified view of the reconstructed resolution target at a distance of 218mm.

part of the resolution target to use is the largest block since it would contain the most pixels thus reducing the error in the result. The size of the largest block is 2.5mm X 2.5mm. By zooming in on this block, it was estimated that the block is 17 pixels X 17 pixels. Due to the noisiness of the image, an error estimate of ± 2 pixels was used which gives a relative error of $\pm 11.76\%$. This gives a pixel size of

$$\frac{2.5mm}{17 \text{ pixels} \pm 11.76\%} = 0.1471mm \pm 11.76\% . \quad (6.7)$$

Thus, the maximum spatial frequency possible is

$$\frac{1lp}{2(0.147mm) \pm 11.76\%} = 3.4 \frac{lp}{mm} \pm 11.76\% . \quad (6.8)$$

This is a range of 3.0 to $3.8 \frac{lp}{mm}$. The theoretical value falls within this range.

The error in the expected spatial resolution must also be evaluated. The maximum spatial frequency is calculated from Eq. (4.2). The magnification M and the distance d are the main factors where a significant error can result. The magnification and its error were given in Eq. (6.3). For the distance d , the error results from the actual placement of the resolution target. The smallest increment on the ruler used in the measurement is 1mm. However, the measurement of the distance was a little difficult and so the error is estimated to be $\pm 2mm$ which gives a relative error of $\pm 0.9174\%$ for a distance of 164mm. Thus, the calculated resolution and its error is

$$\frac{C}{2M\lambda d} = \frac{1024(6.8um)}{2(7.4664 \pm 0.7286\%)(632.8nm)(218mm \pm 0.9174\%)} = 3.380 \frac{lp}{mm} \pm 1.646\% . \quad (6.9)$$

This is a range of 3.324 to $3.436 \frac{lp}{mm}$. This falls well within the measured spatial frequency range given in Eq. (6.8).

The next resolution test was at a distance of 164mm. Actually, the hologram is a double exposure hologram of the resolution target at 164mm and 350mm. The magnified view of the reconstructed resolution target is shown in Fig. 6.6. Visually, the maximum

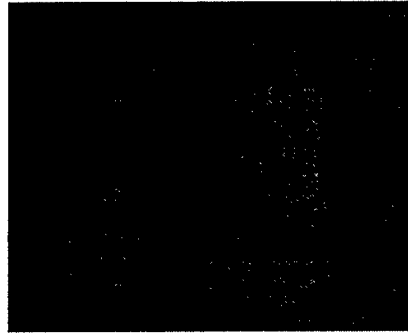


Figure 6.6 Magnified view of the reconstructed resolution target at a distance of $d=164mm$.

spatial frequency resolvable is about Group 1 Element 6. The predicted value was $4.493 \frac{lp}{mm}$, which corresponds to Group 2 Element 2.

As before, the achievable resolution can be estimated by determining the size of the individual pixels that make up the reconstructed image. Once again, the largest block on the resolution target is used to determine the size of the pixels. By zooming in on the reconstructed image, it was estimated that the largest block is 21 pixels X 21 pixels. Again, assuming an error of ± 2 pixels gives a relative error of $\pm 9.524\%$. The resulting pixel size is

$$\frac{2.5mm}{21 \pm 9.524\%} = 0.119mm \pm 9.524\% . \quad (6.10)$$

The corresponding maximum spatial frequency is

$$\frac{1lp}{2(0.119mm) \pm 9.524\%} = 4.2 \frac{lp}{mm} \pm 9.524\% . \quad (6.11)$$

This gives a range of 3.8 to $4.6 \frac{lp}{mm}$. The theoretical value of $4.493 \frac{lp}{mm}$ falls within this range.

The error in the theoretical value should be evaluated as before. The relative error of the distance measurement of $d=164 \pm 2mm$ is $\pm 1.22\%$. The error of the expected value in this case is

$$\frac{C}{2M\lambda d} = \frac{1024(6.8um)}{2(7.4664 \pm 0.7286\%)(632.8nm)(164mm \pm 1.22\%)} = 4.493 \frac{lp}{mm} \pm 1.949\% , \quad (6.12)$$

which is a range of 4.405 to $4.581 \frac{lp}{mm}$. This range falls within the range of the measured value.

The final resolution test is of the resolution target at a distance of 350mm. The reconstruction of the resolution target at 350mm is given in Fig 6.7. A visual inspection

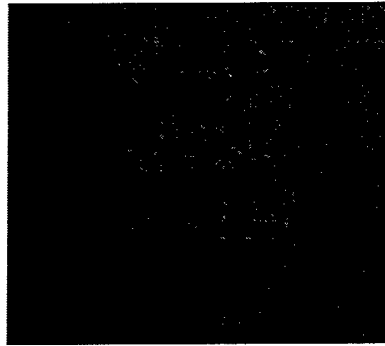


Figure 6.7 Magnified view of the reconstructed resolution target at a distance of 350mm.

of Fig 5.10 reveals that the maximum spatial frequency resolvable is Group 0 Element 5. The expected resolution was $2.105 \frac{lp}{mm}$, which corresponds to about Group 1 Element 1.

Using the same approach as before, the achievable resolution can be estimated by determining the size of the individual pixels that make up the reconstructed image. Again the largest block on the resolution target is used to determine the size of the pixels. By zooming in on the reconstructed image, it was estimated that the largest block is 11 pixels X 11 pixels. Again, assuming an error of ± 2 pixels, the relative error is 18.18%. The resulting pixel size is

$$\frac{2.5mm}{11 \pm 18.18\%} = 0.2273mm \pm 18.18\% . \quad (6.13)$$

The corresponding maximum spatial frequency is

$$\frac{1lp}{2(0.2273mm) \pm 18.18\%} = 2.2 \frac{lp}{mm} \pm 18.18\% . \quad (6.14)$$

Thus, the range is 1.8 to $2.6 \frac{lp}{mm}$. Once again the theoretical value falls within the experimentally determined range.

Following the same analysis as previous, the theoretical value must be modified to take in consideration the errors in the parameters. The relative error in the distance of 350mm is $\pm 0.5714\%$. The range of the theoretical value is therefore

$$\frac{C}{2M\lambda d} = \frac{1024(6.8\mu m)}{2(7.4664 \pm 0.7286\%)(632.8nm)(350mm \pm 0.5714\%)} = 2.105 \frac{lp}{mm} \pm 1.3\% , \quad (6.15)$$

which corresponds to a range of 2.078 to $2.132 \frac{lp}{mm}$. This range falls within the range of the measured value.

6.5 Depth of Field Demonstration

As described in Chapter 4, two approaches were used in demonstrating the depth of field. The first approach was to make a double exposure hologram of the resolution target at two different distances. This is the same hologram used in the previous section for the resolution target at 164mm and 350mm. For the depth of field demonstration, the same hologram was reconstructed at those same distances. The reconstructions are shown in Fig. 6.8. In Fig. 6.8, the larger resolution target is the one located at 164mm while the smaller one is the same resolution target except that it is was moved to a distance of 350mm. In Fig 6.8a, the reconstruction distance, or focus distance, was 164mm. In Fig 6.8b, the reconstruction distance, or focus distance, was 350mm. Referring to Fig 4.4, notice that the depth of field is much smaller when the focus is at 164mm in comparison with when the focus is at 350mm. This effect can be seen in Fig. 6.8. In Fig. 6.8a, the resolution target at 164mm is in focus while at 350mm it is clearly out of focus. On the other hand, in Fig. 6.8b the resolution target at 350mm is in focus,

but also the resolution target at 164mm is still somewhat in focus although noticeably degraded compared to Fig. 6.8a.

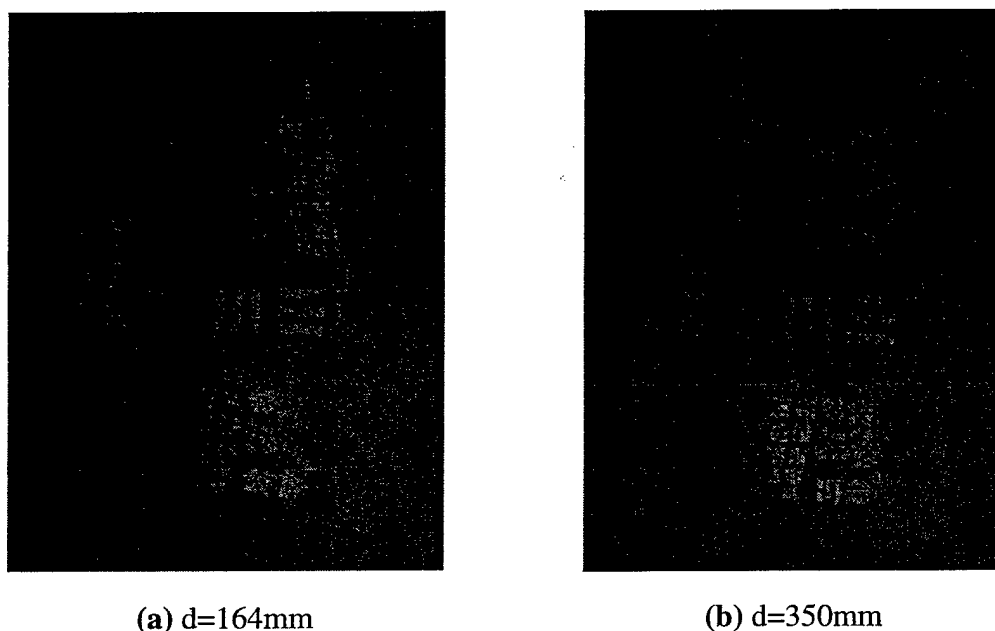


Figure 6.8 Images of the double exposure hologram reconstructed at two different distances: (a) $d=164\text{mm}$. (b) $d=350\text{mm}$.

The other demonstration of depth of field, as described in Chapter 4, is a hologram of a specially designed “ruler”. This ruler was placed at an angle to the hologram film plate. The concept was that different portions of the ruler would come into focus depending on the focus distance. Figure 6.9 shows the reconstructions of the ruler at distances of 73.25mm, 106.8mm, 129.1mm, and 185mm. Using Eqs. (4.8) and (4.9), the corresponding depths of field are as follows:

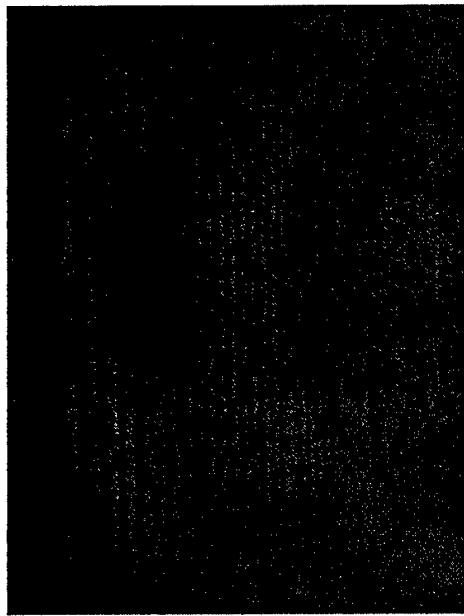
$$\begin{aligned}
 d = 73.25\text{mm} &\Rightarrow \Delta s = \pm 5.547\text{mm} \\
 d = 106.8\text{mm} &\Rightarrow \Delta s = \pm 11.79\text{mm} \\
 d = 129.1\text{mm} &\Rightarrow \Delta s = \pm 17.23\text{mm} \\
 d = 185.0\text{mm} &\Rightarrow \Delta s = \pm 35.38\text{mm}.
 \end{aligned}
 \tag{6.16}$$

The individual lines in the ruler are 1.329mm in width. However, their effective width in terms of distance directly normal to the hologram film plate is 1.299mm. This takes into account the angle of the ruler. This is illustrated in Fig. 6.10. By using the effective width of the lines, the focal plane and depths of field locations can be found roughly by counting the lines.

It is difficult to perceive the actual depths of field from the images in Fig. 6.9 since depth of field is a subjective measure. However, the variations in depth of field are definitely noticeable. The focus distance in Fig 6.9a corresponds to the end of the ruler



(a) $d=73.25\text{mm}$.



(b) $d=106.8\text{mm}$.



(c) $d=129.1\text{mm}$.



(d) $d=185.0\text{mm}$.

Figure 6.9 Reconstruction of ruler at various focus distances: (a) $d=73.25\text{mm}$. (b) $d=106.8\text{mm}$. (c) $d=129.1\text{mm}$. (d) $d=185.0\text{mm}$.

closest to the hologram film plate while in Fig 6.9d the focus is on the end of the ruler farthest from the hologram film plate. As in the double exposure hologram, the larger depth of field at longer distances from the film plate is evident. In Fig. 6.9a, only a small portion of the ruler is in focus while seemingly the entire ruler is in focus in Fig. 6.9d.

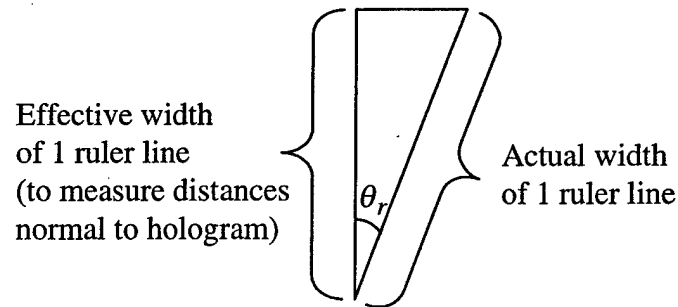


Figure 6.10 Effective width of ruler line.

6.6 Distortion and Noise Considerations

There are three obvious contributions of distortion and noise in the digital reconstructions. They are low pass filtering due to the pixel size, quantization noise, and speckle noise. A description of each of these will now be given.

A major contributor of distortion is the low pass filtering effect caused by the finite size of the pixels in the camera as discussed in Chapter 3. The case of the pixel width being the same as the pixel spacing, as is true with the video camera used in the experiments, was addressed. It was found that the result is a 1.96dB reduction in frequency amplitude for frequencies at half the sampling frequency.

Quantization noise occurs when an analog signal is digitized. The number of bits in the digitization limits the number of levels that are available to represent the signal. This causes the effect known as quantization noise. It is usually estimated that the signal-to-noise ratio attributable to quantization is given by 6dB per bit [Ziemer, et. al., 1983, p. 302]. The digitizer used provides 8 bits per pixel; thus the maximum signal-to-noise ratio is about 48dB.

Laser speckle is another contribution of noise in the digital reconstruction of holograms. Speckle noise occurs due to variations in the surface of the object and film emulsion, film grains, and contaminants in the environment. It is this effect which causes the granularity that appears in reconstructed holograms. Characterization of speckles involves lengthy statistical analysis. The essential results can be found in several texts. For instance, in Goodman, 1996, p. 369, the speckle size σ_s is given to be roughly

$$\sigma_s = \frac{\lambda d}{D}, \quad (6.17)$$

where D is the size of the hologram (apparently assuming use of a circular area of the

hologram). In Vikram, 1992, p. 62, the typical speckle size was given to be

$$\sigma_s = \frac{\lambda d}{r}, \quad (6.18)$$

where r is the radius of the hologram area used in the reconstruction. In Kreis, 1996, p. 37, it was shown that the average size of a speckle for a uniformly scattering square of dimensions $L \times L$ is given by

$$\sigma_s = \frac{\lambda d \pi}{L}. \quad (6.19)$$

The result in Eq. (6.19) seems most applicable for the case of digital holography where a square area of the hologram is typically used as discussed previously.

The importance of Eq. (6.19) is that it suggests noise in the image that is larger than the theoretical resolution capability for a given scanned area of a hologram as derived in Chapter 3. It was shown [Eq. (3.29)] that the maximum resolution for a limited scan area of a hologram is

$$h = \frac{\lambda d}{L}. \quad (6.20)$$

Thus, the average speckle size as given by Eq. (6.19) is larger than the theoretical resolution limit in Eq. (6.20) by a factor of π .

The addition of distortion and noise from each of the described effects will obviously cause degradation in the quality of the reconstructed images. This will make the resolution target used in the experiments more difficult to see for sets of lines that are close to the theoretical resolution limit. Perhaps not so obvious, is the effect on the depth of field. Depth of field is determined by the resolution ability of a system. Better resolution gives a smaller depth of field. Conversely, worse resolution gives a larger depth of field. This is clearly noticeable in Fig 6.9 where the depths of field were larger than anticipated.

7.0 CONCLUSIONS

7.1 Project Results

There were 5 main objectives of this research effort. The first objective was to develop the mathematical theory of digital holography. The result of the theory was that reconstructions are essentially performed with the well-known Fourier transform. Focusing on various parts of an object is achieved by including in the transform an additional phase term.

The second objective was to describe the requirements and effects of digitizing a hologram. Based on the angles between the reference and object waves, it was shown how to determine the numerical aperture of the microscope objective needed to resolve the resulting spatial frequencies of the fringes recorded on the hologram. Additionally, it was shown that the finite size of the pixels in the digitizing camera has a lowpass filtering effect on the reconstructions. Also, it was shown that assembling sub-portions of a hologram into one contiguous hologram required placement accuracies of better than one pixel in order to achieve any benefit.

The third objective was to predict the spatial resolution possible in reconstructions. Based on the mathematical theory, it was shown how much of a hologram must be digitized in order to resolve fine detail of an object. The results indicate that relatively large portions of a hologram must be digitized. For instance, if it is desired to resolve an object that is 0.1mm X 0.1mm in size at a distance of 500mm and assuming the wavelength is 632.8nm, then the hologram must be at least 3.164mm X 3.164mm.

Digitizing an entire hologram of this size would be a formidable task when considering that magnification would probably be necessary. Flatbed scanners have an optical resolution typically around 600 dots per inch (dpi). This is not sufficient to resolve the spatial frequencies present on a typical hologram.

The amount of computer hard disk storage space required for such a hologram is tremendous. If the angles of the object beam and reference beam are assumed to be 30° each, then the spatial frequencies recorded on the hologram will be on the order of 1,580 cycles/mm. Assuming sampling at the Nyquist rate, then 100 megasamples are needed. If the digitization is 8bits/sample, then the necessary storage space is 100 megabytes. Even if an entire hologram can be digitized and stored, a computer system is needed that is capable of processing such large amounts of data in a timely fashion.

The mere storage of the hologram itself as describe above is only part of the memory problem. Since the mathematical operations involved in the reconstructions are complex floating-point operations, the amount of memory needed to process the hologram is even larger. The size of one double-precision complex floating-point number is 16 bytes (8 bytes each for the real and imaginary parts). Thus, the hologram in

our example will now require 1.60 gigabytes of memory to process the entire hologram at one time. The bottom line of the memory issue is can software programs process matrices that are so large, is there enough hard disk storage and RAM memory space, is the time required for processing within tolerable lengths, and, of course, is such a computer system affordable.

The fourth objective was to predict the depth of field in reconstructions. It was shown that the depth of field is dependent on the spatial resolution. It was seen that better resolution implies a smaller depth of field. Conversely, worse resolution implies a larger depth of field.

The final objective was to verify the predictions through experimentation. The experiments performed demonstrated close agreement with the expected spatial resolution. However, the depths of field seen in the reconstructions were larger than anticipated. The primary causes of this result are the distortion and noise as discussed in Chapter 6 and also the subjective nature of perception as to what is considered in-focus.

In conclusion, in time, as the spatial resolution of optical components such as flatbed scanners increases and the capabilities of computer systems improves (which is currently occurring at an incredible rate), then the technique of digital holography will become an increasingly favorable approach for hologram analysis.

7.2 Suggestions for Future Research

There are several areas to suggest for future research. The mosaicking problem is deserving of further attention. The analysis of this research effort was necessarily somewhat simplistic, limiting consideration to only x-y placement locations. There are a number of papers describing and measuring the quality of methods for mosaicking images together. These various methods must be approached cautiously because it is common to warp or otherwise process images to provide an aesthetically appealing mosaic. The consequences of such image processing on the scientific data to be extracted must be considered.

Depth of field lends itself to further research. It has been described [Goodman, 1996, p. 101-107] how a lens can be put in various locations with respect to the hologram plane during reconstruction. If a lens can be simulated and placed at various locations in the digital reconstruction process, then this suggests that the image and object distances can be varied. Thus, the depth of field could possibly be manipulated to some degree.

Perhaps the most obvious is the need for further research is reconstructing images of three dimensions. Each of the images reconstructed in this research effort was two-dimensional. Stereoscopic, tomographic, or other such methods could be applied to attempt to extend this research to true three-dimensional image reconstruction and data extraction. Ultimately, there must be a more direct computational method for

reconstructing in three dimensions rather than to require use of such seemingly cumbersome approaches.

APPENDIX A

Mosaicking Error Analysis Program

The program used to simulate mosaicking for error analysis is named *encsumfp.m*. The following is a listing of the MATLAB (by The MathWorks, Inc.) program code:

```
% Mosaicking Error Analysis
close all
clear all
whitebg('w')
tic
L=32768; % Number of points in one segment of the signal
seg=4; % Number of segments in the signal
f=1/15; % Frequency of the signal
iter=500; % The number of iterations (for statistical purposes)
maxdev=2.5*pi; % This is the maximum phase angle deviation that will be tested
phasenum=5; % The number of steps leading up to the max phase angle deviation
points=1048576; % The signal is zero padded to this number of points
maxx=30; % The encircled power is computed up to this radius (in pixels)
% This set of lines converts the variables from numbers to string for use in labelling
graphs
Lstr=num2str(L);
segstr=num2str(seg);
iterstr=num2str(iter);
phasenumstr=num2str(phasenum+1);
fstr=num2str(f);
pointsstr=num2str(points,8);
maxdevstr=num2str(maxdev);
% These lines create the first segment of the signal
i=1:L;
yseg0=cos(2*pi*f*i);
% These lines perform a Fourier transform, discard half of the result, and then square it
freq=fft(yseg0,points);
freq=freq(1:round(points/2));
freq=(abs(freq)).^2;
% These lines locate the maximum of the energy computed above, determines the
encircled energy of increasingly larger radii, and plots the results
[z,c]=max(freq);
energy=sum(freq);
en=zeros(1,maxx+1);
for p=0:maxx
    en(p+1)=en(p+1)+sum(freq((c-p):(c+p)));
end
prcnten1seg=en/energy;
```



```

plot(0:maxx,prcnten1seg,'k--')
for n=0:phasenum % This for loop increments the maximum phase deviation
    n
    % These lines initialize variables to 0
    totalphi=0;
    en=zeros(1,maxx+1);
    freq=zeros(1,round(points/2));
    for j=1:iter % This for loop iterates the simulation
        segphi=zeros(1,seg); % Initialize segphi to 0
        % This set of lines creates the signal with each segment given a random
        phase with respect to the first segment
        y=yseg0;
        for m=2:seg
            segphi(m)=segphi(m-1)+(rand(1)-0.5)*n/phasenum*maxdev; %
            This computes a random phase for each segment adds it to the
            phase of the previous segment
            yseg=cos(2*pi*f*(i+(m-1)*L)+segphi(m));
            y=[y,yseg];
        end
        % These lines perform a Fourier transform, discard half of the result, and
        then square it
        newfreq=fft(y,points);
        newfreq=newfreq(1:round(points/2));
        newfreq=(abs(newfreq)).^2;
        freq=freq+newfreq; % The result from the above lines is summed for each
        iteration and averaged later
        totalphi=totalphi+sum(segphi); % The phases are summed for each
        iteration and averaged later. The average should be about zero.
    end
    freq=freq/iter;
    % These lines compute the encircled energy for increasingly larger radii
    if n==0
        [z,c]=max(freq); % This determines the location of the maximum energy
        value for the ideal signal and uses it in all subsequent calculations of the
        encircled energy
    end
    energy=sum(freq);
    en=zeros(1,maxx+1);
    for p=0:maxx
        en(p+1)=en(p+1)+sum(freq((c-p):(c+p)));
    end
    prcnten=en/energy;
    % The rest of the lines plot the results and put titles and labels on the graphs
    figure(1)
    hold on

```



```

    grid on
    zoom on
    xlabel('Distance from Center Frequency (pixels)')
    ylabel('Percent of Total Energy')
    strtitle=['Seg L=' Lstr ', # of seg=' segstr ', f=' fstr ', iter=' iterstr ', maxdev='
    maxdevstr ', 0 pad to ' pointsstr];
    title(strtitle)
    plot(0:maxx,prcnten,'k')
    hold off
    figure(n+2)
    plot(0:maxx,prcnten1seg,'k--')
    hold on
    zoom on
    plot(0:maxx,prcnten,'k')
    hold off
    xlabel('Distance from Center Frequency (pixels)')
    ylabel('Percent of Total Energy')
    phidevstr=num2str(n/phasenum*maxdev);
    t=toc
    tstr=num2str(t);
    strtitle=['Seg L=' Lstr ', # of seg=' segstr ', f=' fstr ', iter=' iterstr ', phidev='
    phidevstr ', 0 pad to ' pointsstr];
    title(strtitle)
    avgphi(n+1)=totalphi/iter/seg;
end
t=toc
tstr=num2str(t);
x=maxdev*(0:phasenum)/phasenum;
figure(n+3)
zoom on
plot(x,avgphi,'k')
xlabel('Maximum Deviation from 0 (radians)')
ylabel('Average Phi (radians)')
figure(n+4)
zoom on
plot(y,'k')
figure(n+5)
zoom on
plot(freq,'k')

```


APPENDIX B

Mosaicking Program and Subprograms

The following source code was written in IDL (a product of Research Systems, Inc). The source code is not extensively commented. Its inclusion is intended mainly for reference and documentation. Some of the source code was obtained while at a training class given by Research Systems, Inc. During programming, it was decided that writing one large program was not the best approach because there would be a desire to experiment with variations in the source code. Therefore, different functions were broken out into individual programs so that when changes were made, only one program had to be modified. The programs included in this appendix are the following:

mosaic.pro
error.pro
getseam.pro
mosdims.pro
picklist.pro
read_mos.pro
seaminfo.pro

The source code for *mosaic.pro* is the following:

```
; *****  
;  
; *****  
;  
  
pro zoom_event, event  
  
Widget_Control, event.top, get_uvalue=info, /No_Copy  
widget_control, info.imageid, get_uvalue=images, /no_copy  
possibleEventTypes = [ 'DOWN', 'UP', 'MOTION', 'SCROLL' ]  
thisEvent = possibleEventTypes(event.type)  
IF thisEvent NE 'DOWN' THEN RETURN  
info.mousex=event.x  
info.mousey=event.y  
  
;Change the event handler for the draw widget and turn MOTION events ON.  
Widget_Control, event.id, Event_Pro='zoommove_event', Draw_Motion_Events=1  
  
; Take over color index 1 for the zoom box drawing color. Store the current (r,g,b) values  
; for color index 1 so you can restore the current colors after the zoom box is drawn.  
TVLct, r, g, b, /Get  
info.r_old = r(1)  
info.g_old = g(1)  
info.b_old = b(1)
```



```

widget_control, info.imageid, set_uvalue=images, /no_copy

; Put the info structure back into its storage location.
Widget_Control, event.top, Set_UValue=info, /No_Copy

end ; of zoom_event
; *****

pro zoommove_event,event

; This event handler continuously draws and erases the image box until it receives an UP
; event from the draw widget. Then it turns draw widget motion events OFF and changes
; the event handler for the draw widget back to zoom_PROCESS_EVENTS.

; Get the info structure out of the top-level base.
Widget_Control, event.top, Get_UValue=info, /No_Copy

widget_control, info.imageid, get_uvalue=images, /no_copy

; What type of an event is this?
possibleEventTypes = [ 'DOWN', 'UP', 'MOTION', 'SCROLL' ]
thisEvent = possibleEventTypes(event.type)

IF thisEvent EQ 'UP' THEN BEGIN

    ; If this is an UP event, you need to erase the zoombox, restore the user's color
    ; table, turn motion events OFF, set the draw widget's event handler back to
    ; zoom_PROCESS_EVENTS, and draw the "zoomed" plot in both the draw
    ; widget and the pixmap.

    info.zoomxlow=info.mousex
    info.zoomxhigh=event.x
    info.zoomylow=info.mousey
    info.zoomyhigh=event.y

    ; Make sure the x and y values are ordered correctly
    IF info.zoomxlow GT info.zoomxhigh THEN begin
        info.zoomxlow=event.x
        info.zoomxhigh=info.mousex
    endif
    IF info.zoomylow GT info.zoomyhigh THEN begin
        info.zoomylow=event.y
        info.zoomyhigh=info.mousey
    endif
endif

```



```

; Scale the coordinates
info.zoomxlow=round(info.zoomxlow*info.xscale)
info.zoomxhigh=round(info.zoomxhigh*info.xscale)
info.zoomylow=round(info.zoomylow*info.yscale)
info.zoomyhigh=round(info.zoomyhigh*info.yscale)

WSet, info.widl
Device, Copy = [0, 0, 512, 512, 0, 0, info.boxwid]

; Restore the user's color table
TVLct, info.r_old, info.g_old, info.b_old, 1

; Turn motion events and button events off
Widget_Control, event.id, Draw_Motion_Events=0, draw_button_events=0

display_zoom, info, images
widget_control, info.imageid, set_uvalue=images, /no_copy

; Put the info structure back into its storage location and return.
Widget_Control, event.top, Set_UValue=info, /No_Copy
RETURN

ENDIF ; thisEvent = UP

; Most of the action in this event handler occurs here while we are waiting for an UP
; event to occur. As long as we don't get it, keep erasing the old zoom box and drawing a
; new one.

; Erase the old zoom box.
WSet, info.widl
Device, Copy = [0, 0, 512, 512, 0, 0, info.boxwid]

; Update the dynamic corner of the zoom box to the current cursor location.
newx = event.x
newy = event.y

; Load a color in color index 1 to draw the zoom box with.
TVLct, 255B, 0B, 0B, 1

; Draw the zoom box.
PlotS, [info.mousex, info.mousex, newx, newx, info.mousex], $
      [info.mousey, newy, newy, info.mousey, info.mousey], /Device, Color=1

widget_control, info.imageid, set_uvalue=images, /no_copy

```



```

; Put the info structure back into its storage location.
Widget_Control, event.top, Set_UValue=info, /No_Copy

end ; of zoommove_event
; *****

pro select_event, event

Widget_Control, event.top, get_uvalue=info, /No_Copy
widget_control, info.imageid, get_uvalue=images, /no_copy
possibleEventTypes = [ 'DOWN', 'UP', 'MOTION', 'SCROLL' ]
thisEvent = possibleEventTypes(event.type)
IF thisEvent NE 'DOWN' THEN RETURN

; Obtain the first mouse coordinates
info.mousex=event.x
info.mousey=event.y

; Determine the selected image from the mouse coordinates if the image was not
; selected from the List.
if info.list eq 0 then begin
    for i=0,info.imnum-1 do begin
        if (info.mousex*info.xscale ge images(i).x) and $
            (info.mousex*info.xscale le (images(i).x+info.subxsize)) and $
            (info.mousey*info.yscale ge images(i).y) and $
            (info.mousey*info.yscale le (images(i).y+info.subysize)) then $
            info.imsel=i ; The HIGHEST image # in the area will be selected
    endfor
endif

; Take over color index 1 for the zoom box drawing color. Store the current (r,g,b) values
; for color index 1 so you can restore the current colors after the zoom box is drawn.
TVLct, r, g, b, /Get
info.r_old = r(1)
info.g_old = g(1)
info.b_old = b(1)

; Load a color in color index 1 to draw the zoom box with.
TVLct, 255B, 0B, 0B, 1

; Change the event handler for the draw widget and turn MOTION events ON.
Widget_Control, event.id, Event_Pro='move_event', Draw_Motion_Events=1

widget_control, info.imageid, set_uvalue=images, /no_copy

```



```

; Put the info structure back into its storage location.
Widget_Control, event.top, Set_UValue=info, /No_Copy

end ; of select_event
; *****

pro move_event,event

; This event handler continuously draws and erases the image box until it receives an UP
; event from the draw widget. Then it turns draw widget motion events OFF and changes
; the event handler for the draw widget back to zoom_PROCESS_EVENTS.

; Get the info structure out of the top-level base.
Widget_Control, event.top, Get_UValue=info, /No_Copy

widget_control, info.imageid, get_uvalue=images, /no_copy

; What type of an event is this?
possibleEventTypes = [ 'DOWN', 'UP', 'MOTION', 'SCROLL' ]
thisEvent = possibleEventTypes(event.type)

IF thisEvent EQ 'UP' THEN BEGIN

    ; If this is an UP event, you need to erase the zoombox, restore the user's color
    ; table, turn motion events OFF, set the draw widget's event handler back to
    ; zoom_PROCESS_EVENTS, and draw the "zoomed" plot in both the draw
    ; widget and the pixmap.

    ; store the new coordinates for the selected image
    delx = (event.x-info.mousex)*info.xscale
    dely = (event.y-info.mousey)*info.yscale
    images(info.imsel).x=images(info.imsel).x+delx
    images(info.imsel).y=images(info.imsel).y+dely

    ; Erase the zoombox one final time by copying the plot from the pixmap.
    WSet, info.wid1
    Device, Copy = [0, 0, 512, 512, 0, 0, info.boxwid]

    ; Restore the user's color table.
    TVLct, info.r_old, info.g_old, info.b_old, 1

    ; Turn motion events and button events off
    Widget_Control, event.id, Draw_Motion_Events=0, draw_button_events=0

```



```

display_mosaic, info, images
widget_control, info.imageid, set_uvalue=images, /no_copy

; Put the info structure back into its storage location and return.
Widget_Control, event.top, Set_UValue=info, /No_Copy
RETURN

ENDIF ; thisEvent = UP

; Most of the action in this event handler occurs here while we are waiting for an UP
; event to occur. As long as we don't get it, keep erasing the old zoom box and drawing a
; new one.

; Erase the old zoom box.
WSet, info.widl
Device, Copy = [0, 0, 512, 512, 0, 0, info.boxwid]

; Update the dynamic corner of the zoom box to the current cursor location.
delx = event.x-info.mousex
dely = event.y-info.mousey

; Draw the zoom box.
PlotS, [round(images(info.imsel).x/info.xscale+delx), $
      round((images(info.imsel).x+info.subxsize)/info.xscale+delx), $
      round((images(info.imsel).x+info.subxsize)/info.xscale+delx), $
      round(images(info.imsel).x/info.xscale+delx), $
      round(images(info.imsel).x/info.xscale+delx)], $
      [round(images(info.imsel).y/info.yscale+dely), $
      round(images(info.imsel).y/info.yscale+dely), $
      round((images(info.imsel).y+info.subysize)/info.yscale+dely), $
      round((images(info.imsel).y+info.subysize)/info.yscale+dely), $
      round(images(info.imsel).y/info.yscale+dely)], /Device, Color=1

widget_control, info.imageid, set_uvalue=images, /no_copy

; Put the info structure back into its storage location.
Widget_Control, event.top, Set_UValue=info, /No_Copy

end ; of move_event
; *****

; Display Mosaic procedure
pro display_mosaic, info, images

mosaic=bytarr(info.xmos,info.ymos)

```



```

for i=0,info.imnum-1 do begin
    mosaic(images(i).x,images(i).y)=images(i).img
endfor
wset, info.widl
erase
tvscl, congrid(mosaic,512,512)
widget_control,info.mosaicid,set_uvalue=mosaic, /no_copy
end ; of Display Mosaic
; *****

; Display Zoom procedure
pro display_zoom, info, images

mosaic=bytarr(info.xmos,info.ymos)
for i=0,info.imnum-1 do begin
    mosaic(images(i).x,images(i).y)=images(i).img
endfor
wset, info.widl
erase
tvscl,congrid(mosaic(info.zoomxlow:info.zoomxhigh,info.zoomylow:info.zoomyhigh), $
    512,512)
widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy
end ; of Display Zoom
; *****

PRO mosaic_Event, event

widget_control,event.top,get_uvalue=info, /No_Copy
widget_control,event.id,get_uvalue=value
widget_control,info.mosaicid,get_uvalue=mosaic, /No_Copy
widget_control,info.imageid,get_uvalue=images, /No_Copy

CASE value OF
    'MOSDIM' : begin
        widget_control,event.top,tlb_get_offset=offsets ;obtain mosaic X & Y size
        stuff=mosdims(offsets(0)+50,offsets(1)+75, info.xmos, info.ymos, $
            info.subxsize, info.subysize,info.subnum)
        if stuff.xsize ne 0 then begin
            info.xmos=stuff.xsize
            info.ymos=stuff.ysize
            info.subnum=stuff.subnum
            info.subxsize=stuff.subxsize
            info.subysize=stuff.subysize
            mosaic=bytarr(info.xmos,info.ymos)
            images={img:bytarr(info.subxsize,info.subysize),x:0,y:0,name:"}

```



```

        images=replicate(images,info.subnum)
        info.xscale=info.xmos/512.
        info.yscale=info.ymos/512.
        Window, /Free, XSize=512, YSize=512, /Pixmap
        info.boxwid = !D.Window
        widget_control, info.mosaicid, set_uvalue=mosaic, /no_copy
        widget_control,info.imageid,set_uvalue=images, /no_copy
        widget_control,event.top,set_uvalue=info, /No_Copy
    endif
end
'LOAD' :begin
    widget_control,event.top,tlb_get_offset=offsets ;load images
    pick=dialog_pickfile(/read, path=info.impath, get_path=impath)
    info.impath=impath

    ;determine the image type
    pick=strtolower(pick)
    type=strmid(pick,strlen(pick)-3,3)
    if type eq 'byt' then imtype=0
    if type eq 'tif' then imtype=1
    if type eq 'byt' or type eq 'tif' then begin
        images(info.imnum).name=pick
        info.imtype=imtype
        imnum=info.imnum
        images(info.imnum).img=read_mos(images(info.imnum).name, $
            info.imtype, info.subxsize, info.subysize, info.wid1)
        info.imnum=info.imnum+1
        display_mosaic, info, images
        widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy
        widget_control, info.imageid, set_uvalue=images, /no_copy
        widget_control, event.top, set_uvalue=info, /No_Copy
    endif else nothing=error(offsets(0)+50,offsets(1)+75)

end
'SELECT' :begin
    widget_control, info.imageid, draw_Button_Events=1, $
        event_pro='select_event'
    info.list=0
    wset,info.boxwid
    device, copy = [0, 0, 512, 512, 0, 0, info.wid1]
    widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy
    widget_control, info.imageid, set_uvalue=images, /no_copy
    Widget_Control, event.top, Set_UValue=info, /No_Copy
end
'LIST' :begin

```



```

widget_control,event.top,tlb_get_offset=offsets ;obtain new coordinates
info.list=1
names=strarr(info.subnum)
for i=0,info.subnum-1 do begin
    names(i)=images(i).name
endfor
pick=picklist(offsets(0)+50,offsets(1)+75,info.subnum,info.imsel,names)
info.imsel=pick
widget_control, info.imageid, draw_Button_Events=1, $
    event_pro='select_event'

; Make a copy of the current display
wset,info.boxwid
device, copy = [0, 0, 512, 512, 0, 0, info.wid1]

; Take over color index 1 for the zoom box drawing color. Store the
; current (r,g,b) values for color index 1 so you can restore the current
; colors after the zoom box is drawn.
TVLct, r, g, b, /Get
info.r_old = r(1)
info.g_old = g(1)
info.b_old = b(1)

; Load a color in color index 1 to draw the zoom box with.
TVLct, 255B, 0B, 0B, 1

; Set the current window to the display.
wset,info.wid1

; Draw the box.
PlotS, [round(images(info.imsel).x/info.xscale), $
    round((images(info.imsel).x+info.subxsize)/info.xscale), $
    round((images(info.imsel).x+info.subxsize)/info.xscale), $
    round(images(info.imsel).x/info.xscale), $
    round(images(info.imsel).x/info.xscale)], $
    [round(images(info.imsel).y/info.yscale), $
    round(images(info.imsel).y/info.yscale), $
    round((images(info.imsel).y+info.subysize)/info.yscale), $
    round((images(info.imsel).y+info.subysize)/info.yscale), $
    round(images(info.imsel).y/info.yscale)], /Device, Color=1

widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy
widget_control, info.imageid, set_uvalue=images, /no_copy
Widget_Control, event.top, Set_UValue=info, /No_Copy
end

```



```

'SHOW' :begin
    mosaic=mosaic*0.
    mosaic(images(info.imsel).x,images(info.imsel).y)= $
        images(info.imsel).img
    wset, info.widl
    erase
    tvscl, congrid(mosaic,512,512)
    widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy
    widget_control, info.imageid, set_uvalue=images, /no_copy
    Widget_Control, event.top, Set_UValue=info, /No_Copy
end
'SEAM' :begin
    widget_control,event.top,tlb_get_offset=offsets ;obtain new coordinates
    info.seam=getseam(offsets(0)+50,offsets(1)+75)
    nothing=seaminfo(offsets(0)+50, offsets(1)+75)
    cursor, corx, cory, /down, /device
    info.corx=round(corx*info.xscale)
    info.cory=round(cory*info.yscale)
    print,'corx=',info.corx,'cory=',info.cory
    widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy
    widget_control, info.imageid, set_uvalue=images, /no_copy
    Widget_Control, event.top, Set_UValue=info, /No_Copy
end
'CORMATCH' :begin
    widget_control,event.top,tlb_get_offset=offsets
    cormax=0
    xloc=0
    yloc=0

    ; corx & cory are the user selected cursor points. The user selects a point
    ; on the mosaic for matching. The user selects whether to use a horizontal
    ; or vertical line for matching. The length of the line is defined by the
    ; variable length. The result is stored in corvec1. The 2 for loops take this
    ; line and looks for the best match within the selected image defined by
    ; info.imsel. The variable range defines the number of lines (either
    ; horizontal or vertical) to search through within the selected image. The
    ; indices i and j define the coordinates within the selected image to search.
    ; Equally sized lines at these coordinates are iteratively stored in corvec2
    ; and compared with corvec1. xloc & yloc are the coordinates of the best
    ; match.
    length=256; line length to use in matching
    range=100
    if info.seam eq 0 then begin; horizontal matching
        corvec1=mosaic(info.corx:(info.corx+(length-1)),info.cory)
        for i=(info.subxsize/2-range),(info.subxsize/2+range) do begin

```



```

        for j=0,100 do begin; # of lines to search
            corvec2=images(info.imsel).img(i:(i+(length-1)),j)
            coramount=correlate(corvec1,corvec2)
            if coramount gt cormax then begin
                cormax=coramount
                xloc=i
                yloc=j
            endif
        endfor
    endfor
endif else begin; vertical matching
    corvec1=mosaic(info.corx,info.cory:(info.cory+(length-1)))
    for i=0,100 do begin; # of lines to search
        for j=(info.subysize/2-range),(info.subysize/2+range) $
            do begin
                corvec2=images(info.imsel).img(i,j:(j+(length-1)))
                coramount=correlate(corvec1,corvec2)
                if coramount gt cormax then begin
                    cormax=coramount
                    xloc=i
                    yloc=j
                endif
            endfor
        endfor
    endelse
    print,'xloc=',xloc,'yloc=',yloc
    if (info.corx-xloc) lt 0 or (info.cory-yloc) lt 0 $
        or (info.corx+(info.subxsize-xloc)) gt info.xmos $
        or (info.cory+(info.subysize-yloc)) gt info.ymos then begin
        nothing=error(offsets(0)+50,offsets(1)+75)
    endif else begin
        ; moves selected image to the matching location
        images(info.imsel).x=info.corx-xloc
        images(info.imsel).y=info.cory-yloc
    endelse

    mosaic=bytarr(info.xmos,info.ymos)
    for i=0,info.imnum-1 do begin
        mosaic(images(i).x,images(i).y)=images(i).img
    endfor
    widget_control,info.mosaicid,set_uvalue=mosaic,/No_Copy
    widget_control,info.imageid,set_uvalue=images,/no_copy
    Widget_Control,event.top,Set_UValue=info,/No_Copy
end
'DISPLAY':begin

```



```

display_mosaic, info, images
widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy
widget_control, info.imageid, set_uvalue=images, /no_copy
Widget_Control, event.top, Set_UValue=info, /No_Copy
end
'ZOOM' :begin
widget_control, info.imageid, draw_Button_Events=1, $
    event_pro='zoom_event'
wset,info.boxwid
device, copy = [0, 0, 512, 512, 0, 0, info.wid1]
widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy
widget_control, info.imageid, set_uvalue=images, /no_copy
Widget_Control, event.top, Set_UValue=info, /No_Copy
end
'DISPZOOM' :begin
display_zoom, info, images
widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy
widget_control, info.imageid, set_uvalue=images, /no_copy
Widget_Control, event.top, Set_UValue=info, /No_Copy
end
'SAVE' :begin
mosaic=reverse(mosaic,2)
pick=dialog_pickfile(/write, path=info.impath, get_path=impath)
info.impath=impath
tiff_write,pick,mosaic
widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy
widget_control, info.imageid, set_uvalue=images, /no_copy
Widget_Control, event.top, Set_UValue=info, /No_Copy
end
'SAVEDAT' :begin
stuff=dialog_pickfile(/write, path='c:\', filter='*.dat')
save, mosaic, images, info, filename=stuff
widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy
widget_control, info.imageid, set_uvalue=images, /no_copy
Widget_Control, event.top, Set_UValue=info, /No_Copy
end
'RESTORE' :begin
wid1=info.wid1
stuff=dialog_pickfile(/read, path='c:\', filter='*.dat')
restore, stuff
info.wid1=wid1
Window, /Free, XSize=512, YSize=512, /Pixmap
info.boxwid = !D.Window
display_mosaic, info, images
widget_control,info.mosaicid,set_uvalue=mosaic, /No_Copy

```



```

        widget_control, info.imageid, set_uvalue=images, /no_copy
        Widget_Control, event.top, Set_UValue=info, /No_Copy
    end
    'EXIT':Widget_Control, event.top, /Destroy
ENDCASE

END ; of mosaic_Event

; *****

PRO mosaic

; images are transferred from bottom to top, i.e. the row with a 0 subscript is drawn on the
; bottom
!order=0

device, retain=2
base = Widget_Base(Title='Image Mosaicking', Column=2)
drawbase=widget_base(base,column=2)
btnbase=widget_base(base,column=1)
btn1=widget_button(btnbase,value='Define Mosaic Dimensions',uvalue='MOSDIM')
btn2=widget_button(btnbase,value='Load Image',uvalue='LOAD')
btn3=widget_button(btnbase,value='Select Image with Mouse',uvalue='SELECT')
btn4=widget_button(btnbase,value='Select Image from List',uvalue='LIST')
btn5=widget_button(btnbase,value='Show Current Image',uvalue='SHOW')
btn11=widget_button(btnbase,value='Define Seam',uvalue='SEAM')
btn12=widget_button(btnbase,value='Match Image',uvalue='CORMATCH')
btn6=widget_button(btnbase,value='Display Mosaic',uvalue='DISPLAY')
btn7=widget_button(btnbase,value='Zoom',uvalue='ZOOM')
btn10=widget_button(btnbase,value='Display Mosaic Zoom Area',uvalue='DISPZOOM')
btn8=widget_button(btnbase,value='Save Mosaic (TIF)',uvalue='SAVE')
btn13=widget_button(btnbase,value='Save Progress',uvalue='SAVEDAT')
btn14=widget_button(btnbase,value='Restore Progress',uvalue='RESTORE')
btn9=widget_button(btnbase,value='Exit',uvalue='EXIT')
draw1=widget_draw(drawbase,xsize=512,ysize=512)
Widget_Control, base, /Realize
widget_control,draw1,get_value=wid1
info={xmos:2048,ymos:2048,zoomxhigh:50,zoomxlow:40,zoomyhigh:50, $
      zoomylow:40,wid1:wid1,mosaicid:drawbase,imtype:0,imageid:draw1, $
      disptype:1,imnum:0,imsel:0,subnum:4,subxsize:1024,subysize:1024,seam:0, $
      corx:0,cory:0,r_old:255b,g_old:255b,b_old:255b,mousex:0,mousey:0, $
      boxwid:0b,xscale:1.0,yscale:1.0,list:0,impath:'c:\'}
mosaic=0b
images=0b
loadct,0

```



```

wset,wid1
erase
widget_control,base,set_uvalue=info, /No_Copy
widget_control,drawbase,set_uvalue=mosaic, /No_Copy
widget_control,draw1,set_uvalue=images, /No_Copy
XManager, 'mosaic', base, Event='mosaic_Event'
END ; of mosaic

```

```

, *****
,
, *****
,

```

The source code for *error.pro* is the following:

```

, *****
,
, *****
,

```

Pro error_Event, event

```

name = Tag_Names(event, /Structure_Name)
IF name EQ 'WIDGET_BUTTON' THEN BEGIN
    Widget_Control, event.top, /Destroy
endif
End ; of error_Event

```

```

, *****
,

```

Function error, xoff, yoff

```

base = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, $
    Title='Program Error!', xsize=150)
ok = Widget_Button(base, Value='OK')
Widget_Control, base, /Realize
XManager, 'error', base, /Modal
return,0
End ; of error

```

```

, *****
,
, *****
,

```

The source code for *getseam.pro* is the following:

```

, *****
,
, *****
,

```


Pro getseam_Event, event

name = Tag_Names(event, /Structure_Name)

IF name EQ 'WIDGET_BUTTON' THEN BEGIN

Widget_Control, event.top, Get_UValue=info, /No_Copy

Widget_Control, event.id, Get_Value=test

CASE test OF

; 0=horizontal, 1=vertical

'HORIZONTAL' : Begin

data = 0

Handle_Value, info.handle, data, /Set, /No_Copy

Widget_Control, event.top, /Destroy

End

'VERTICAL' : Begin

data = 1

Handle_Value, info.handle, data, /Set, /No_Copy

Widget_Control, event.top, /Destroy

End

ENDCASE

ENDIF

End ; of getseam_Event

; *****

Function getseam, xoff, yoff

base = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, \$

Title='Click on Seam Direction')

horiz = Widget_Button(base, Value='HORIZONTAL')

vert = Widget_Button(base, Value='VERTICAL')

Widget_Control, base, /Realize

handle = handle_create()

info={handle:handle}

Widget_Control, base, Set_UValue=info, /No_Copy

XManager, 'getseam', base, /Modal

Handle_Value, handle, data, /No_Copy

Handle_Free, handle

Return, data

End ; of getseam

; *****
;
; *****

The source code for *mosdims.pro* is the following:

```

; *****
; *****
; *****

Pro mosdims_Event, event

name = Tag_Names(event, /Structure_Name)
IF name EQ 'WIDGET_BUTTON' THEN BEGIN
    Widget_Control, event.top, Get_UValue=info, /No_Copy
    Widget_Control, event.id, Get_Value=test

    CASE test OF
        'CANCEL'      : Begin
            data={xsize:0}
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            End
        'ACCEPT'      : Begin
            Widget_Control, info.xsize_id, Get_Value=xsize
            Widget_Control, info.ysize_id, Get_Value=ysize
            Widget_Control, info.subxsize_id, Get_Value=subxsize
            Widget_Control, info.subysize_id, Get_Value=subysize
            Widget_Control, info.subnum_id, Get_Value=subnum
            data = {xsize:xsize, ysize:ysize, subnum:subnum, $
                    subxsize:subxsize, subysize:subysize}
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            End
    ENDCASE

ENDIF
END ; mosdims_Event
; *****

Function mosdims, xoff, yoff, currentx, currenty, currentsubx, currentsuby, $
    currentsubnum

base = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, $
    Title='Enter Mosaic Information')
mosbase = Widget_Base(base, Column=1)
filelabel = Widget_Label(mosbase, Value='Input Mosaic Size')
xsize_id = CW_Field(mosbase, Title='X Size: ', Value=currentx, $
    /Integer, /Return_Events)
ysize_id = CW_Field(mosbase, Title='Y Size: ', Value=currenty, $

```



```

        /Integer, /Return_Events)
subnumbase=Widget_Base(base, Column=1)
filelabel2 = Widget_Label(subnumbase, Value='Input # of Sub-Images')
subnum_id = CW_Field(subnumbase, Title='Number:', Value=currentsubnum, /Integer, $
/Return_Events)
subsizebase=Widget_Base(base, Column=1)
filelabel3 = Widget_Label(subsizebase, Value='Input Sub-Image Size')
subxsize_id = CW_Field(subsizebase, Title='X Size: ', Value=currentsubx, /Integer, $
/Return_Events)
subysize_id = CW_Field(subsizebase, Title='Y Size: ', Value=currentsuby, /Integer, $
/Return_Events)
btnbase = Widget_Base(base, Row=1, /Frame)
cancel = Widget_Button(btnbase, Value='CANCEL')
done = Widget_Button(btnbase, Value='ACCEPT')
Widget_Control, base, /Realize
handle = handle_create()
info = {xsize_id:xsize_id, ysize_id:ysize_id, handle:handle, currentx:currentx, $
currenty:currenty, subnum_id:subnum_id, $ subxsize_id:subxsize_id, $
subysize_id:subysize_id}
Widget_Control, base, Set_UValue=info, /No_Copy
XManager, 'mosdims', base, /Modal

;Must use handle because all widgets are destroyed after Xmanager.
Handle_Value, handle, data, /No_Copy

Handle_Free, handle
Return, data
End ; of mosdims

```

```

; *****
; *****
;

```

The source code for *picklist.pro* is the following:

```

; *****
; *****
;

```

Pro picklist_Event, event

```

name = Tag_Names(event, /Structure_Name)
IF name EQ 'WIDGET_LIST' THEN BEGIN
    Widget_Control, event.top, Get_UValue=info, /No_Copy
    data=event.index
    Handle_Value, info.handle, data, /Set, /No_Copy

```



```

Widget_Control, event.top, set_UValue=info, /No_Copy
Widget_Control, event.top, /Destroy
ENDIF
END ; of picklist_Event
; *****

Function picklist, xoff, yoff, number, imsel, names

base = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, Title='Image List')
list=widget_list(base, value=names, ysize=number)
Widget_Control, base, /Realize
handle = handle_create()
info={ handle:handle,listid:list}
Widget_Control, base, Set_UValue=info, /No_Copy
widget_control, list, set_uvalue=names, /no_copy
XManager, 'picklist', base, /Modal

;Must use handle because all widgets are destroyed after Xmanager.
Handle_Value, handle, data, /No_Copy

Handle_Free, handle
Return, data
End ; of picklist

; *****
; *****
; *****

```

The source code for *read_mos.pro* is the following:

```

; *****
; *****
; *****

Function read_mos, file, imtype, m, n, wid1

print,m,n

;read byte
if imtype eq 0 then begin
    openr,1,file
    newimg=bytarr(m,n)
    readu,1,newimg
    close,1
endif

```



```

;read tiff
if imtype eq 1 then begin
    newimg=tiff_read(file)
endif

newimg=reverse(newimg,2); reverses the rows
return, newimg
end ; of read_mos

```

```

; *****
; *****
;

```

The source code for *seaminfo.pro* is the following:

```

; *****
; *****
;

```

Pro seaminfo_Event, event

```

name = Tag_Names(event, /Structure_Name)
IF name EQ 'WIDGET_BUTTON' THEN BEGIN
    Widget_Control, event.top, /Destroy
endif
End ; of seaminfo_Event

```

```

; *****
;

```

Function seaminfo, xoff, yoff

```

base = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, $
    Title='Define Seam Search')
filebase = Widget_Base(base, Column=1)
filelabel= Widget_Label(base, Value='Click Mouse At Correlation Line')
ok = Widget_Button(base, Value='OK')
Widget_Control, base, /Realize
XManager, 'seaminfo', base, /Modal
return,0
End ; of seaminfo

```

```

; *****
; *****
;

```


APPENDIX C

Digital Reconstruction Program and Subprograms

As in Appendix B, the following source code was written in IDL. The source code is not extensively commented. Its inclusion is intended mainly for reference and documentation. Some of the source code was obtained while at a training class given by Research Systems, Inc. During programming, it was decided that writing one large program was not the best approach because there would be a desire to experiment with variations in the source code. Therefore, different functions were broken out into individual programs so that when changes were made, only one program had to be modified. The programs included in this appendix are the following:

```
dhdouble.pro
getd.pro
getdim.pro
getmisc.pro
imcutarr.pro
linelog.pro
orient.pro
read_dh.pro
```

The source code for *dhdouble.pro* is the following:

```
; *****
; *****
```

```
pro mask_event, event
```

```
Widget_Control, event.top, get_uvalue=info, /No_Copy
widget_control, info.holoid, get_uvalue=holo, /no_copy
possibleEventTypes = [ 'DOWN', 'UP', 'MOTION', 'SCROLL' ]
thisEvent = possibleEventTypes(event.type)
IF thisEvent NE 'DOWN' THEN RETURN
info.mousex=event.x
info.mousey=event.y
```

```
; Change the event handler for the draw widget and turn MOTION events ON.
Widget_Control, event.id, Event_Pro='maskmove_event', Draw_Motion_Events=1
```

```
; Take over color index 1 for the zoom box drawing color. Store the current (r,g,b) values
; for color index 1 so you can restore the current colors after the zoom box is drawn.
TVLct, r, g, b, /Get
info.r_old = r(1)
info.g_old = g(1)
```



```

info.b_old = b(1)

widget_control, info.holoid, set_uvalue=holo, /no_copy

; Put the info structure back into its storage location.
Widget_Control, event.top, Set_UValue=info, /No_Copy

end ; of mask_event
; *****

pro maskmove_event,event

; This event handler continuously draws and erases the image box until it receives an UP
; event from the draw widget. Then it turns draw widget motion events OFF and changes
; the event handler for the draw widget back to zoom_PROCESS_EVENTS.

; Get the info structure out of the top-level base.
Widget_Control, event.top, Get_UValue=info, /No_Copy

widget_control, info.drawlid, get_uvalue=recon, /no_copy

; What type of an event is this?
possibleEventTypes = [ 'DOWN', 'UP', 'MOTION', 'SCROLL' ]
thisEvent = possibleEventTypes(event.type)

IF thisEvent EQ 'UP' THEN BEGIN

    ; If this is an UP event, you need to erase the zoombox, restore the user's color
    ; table, turn motion events OFF, set the draw widget's event handler back to
    ; zoom_PROCESS_EVENTS, and draw the "zoomed" plot in both the draw
    ; widget and the pixmap.

    ; Turn motion events and button events off.
    Widget_Control, event.id, Draw_Motion_Events=0, draw_button_events=0

    ; Restore the user's color table.
    TVLct, info.r_old, info.g_old, info.b_old, 1

    info.maskxlow=info.mousex
    info.maskxhigh=event.x
    info.maskylo=info.mousey
    info.maskyhigh=event.y

    ; Make sure the x and y values are ordered correctly
    IF info.maskxlow GT info.maskxhigh THEN begin

```



```

        info.maskxlow=event.x
        info.maskxhigh=info.mousex
    endif
    IF info.maskylo w GT info.maskyhigh THEN begin
        info.maskylo w=event.y
        info.maskyhigh=info.mousey
    endif

; Scale the coordinates
info.maskxlow=round(info.maskxlow*info.xscale)
info.maskxhigh=round(info.maskxhigh*info.xscale)
info.maskylo w=round(info.maskylo w*info.yscale)
info.maskyhigh=round(info.maskyhigh*info.yscale)

; Display the masked out recon
recon(info.maskxlow:info.maskxhigh,info.maskylo w:info.maskyhigh)=0
if info.disptype eq 1 then begin
    wset,info.wid1
    erase
    tvscl,congrid(recon,512,512)
    wset,info.wid2
    erase
    tvscl,recon(info.zoomxlow:info.zoomxhigh,$
        info.zoomylo w:info.zoomyhigh)
endif
if info.disptype eq 2 then begin
    wset,info.wid1
    erase
    tvscl,congrid(alog(recon > 1e-6),512,512)
    wset,info.wid2
    erase
    tvscl,alog(recon(info.zoomxlow:info.zoomxhigh,$
        info.zoomylo w:info.zoomyhigh) > 1e-6)
endif

widget_control, info.draw lid, set_uvalue=recon, /no_copy

; Put the info structure back into its storage location and return.
Widget_Control, event.top, Set_UValue=info, /No_Copy
RETURN

```

ENDIF ; thisEvent = UP

; Most of the action in this event handler occurs here while we are waiting for an UP
; event to occur. As long as we don't get it, keep erasing the old zoom box and drawing a


```

; new one.

; Erase the old zoom box.
WSet, info.wid1
Device, Copy = [0, 0, 512, 512, 0, 0, info.boxwid]

; Update the dynamic corner of the zoom box to the current cursor location.
newx = event.x
newy = event.y

; Load a color in color index 1 to draw the zoom box with.
TVLct, 255B, 0B, 0B, 1

; Draw the zoom box.
PlotS, [info.mousex, info.mousex, newx, newx, info.mousex], $
      [info.mousey, newy, newy, info.mousey, info.mousey], /Device, Color=1

widget_control, info.drawlid, set_uvalue=recon, /no_copy

; Put the info structure back into its storage location.
Widget_Control, event.top, Set_UValue=info, /No_Copy

end ; of maskmove_event
; *****

pro zoom_event, event

Widget_Control, event.top, get_uvalue=info, /No_Copy
widget_control, info.holoid, get_uvalue=holo, /no_copy
possibleEventTypes = [ 'DOWN', 'UP', 'MOTION', 'SCROLL' ]
thisEvent = possibleEventTypes(event.type)
IF thisEvent NE 'DOWN' THEN RETURN
info.mousex=event.x
info.mousey=event.y

; Change the event handler for the draw widget and turn MOTION events ON.
Widget_Control, event.id, Event_Pro='zoommove_event', Draw_Motion_Events=1

; Take over color index 1 for the zoom box drawing color. Store the current (r,g,b) values
; for color index 1 so you can restore the current colors after the zoom box is drawn.
TVLct, r, g, b, /Get
info.r_old = r(1)
info.g_old = g(1)
info.b_old = b(1)

```



```

widget_control, info.holoid, set_uvalue=holo, /no_copy

; Put the info structure back into its storage location.
Widget_Control, event.top, Set_UValue=info, /No_Copy

end ; of zoom_event
; *****

pro zoommove_event,event

; This event handler continuously draws and erases the image box until it receives an UP
; event from the draw widget. Then it turns draw widget motion events OFF and changes
; the event handler for the draw widget back to zoom_PROCESS_EVENTS.

; Get the info structure out of the top-level base.
Widget_Control, event.top, Get_UValue=info, /No_Copy

widget_control, info.draw1id, get_uvalue=recon, /no_copy

; What type of an event is this?
possibleEventTypes = [ 'DOWN', 'UP', 'MOTION', 'SCROLL' ]
thisEvent = possibleEventTypes(event.type)

IF thisEvent EQ 'UP' THEN BEGIN

    ; If this is an UP event, you need to erase the zoombox, restore the user's color
    ; table, turn motion events OFF, set the draw widget's event handler back to
    ; zoom_PROCESS_EVENTS, and draw the "zoomed" plot in both the draw
    ; widget and the pixmap.

    ; Turn motion events and button events off.
    Widget_Control, event.id, Draw_Motion_Events=0, draw_button_events=0

    ; Restore the user's color table.
    TVLct, info.r_old, info.g_old, info.b_old, 1

    info.zoomxlow=info.mousex
    info.zoomxhigh=event.x
    info.zoomylow=info.mousey
    info.zoomyhigh=event.y

    ; Make sure the x and y values are ordered correctly.
    IF info.zoomxlow GT info.zoomxhigh THEN begin
        info.zoomxlow=event.x
        info.zoomxhigh=info.mousex
    
```



```

endif
IF info.zoomylo w GT info.zoomyhigh THEN begin
    info.zoomylo w=event.y
    info.zoomyhigh=info.mousey
endif

; Scale the coordinates
info.zoomxlo w=round(info.zoomxlo w*info.xscale)
info.zoomxhigh=round(info.zoomxhigh*info.xscale)
info.zoomylo w=round(info.zoomylo w*info.yscale)
info.zoomyhigh=round(info.zoomyhigh*info.yscale)
print,info

; Display the zoomed recon
wset, info.wid1
erase
wset, info.wid2
erase
if info.disptype eq 1 then begin
    wset,info.wid1
    tvscl,congrid(recon,512,512)
    wset,info.wid2
    tvscl,congrid(recon(info.zoomxlo w:info.zoomxhigh,$
        info.zoomylo w:info.zoomyhigh),150,150)
endif
if info.disptype eq 2 then begin
    wset,info.wid1
    tvscl,congrid(alog(recon > 1e-6),512,512)
    wset,info.wid2
    tvscl,congrid(alog(recon(info.zoomxlo w:info.zoomxhigh,$
        info.zoomylo w:info.zoomyhigh) > 1e-6),150,150)
endif

widget_control, info.drawlid, set_uvalue=recon, /no_copy

; Put the info structure back into its storage location and return.
widget_Control, event.top, Set_UValue=info, /No_Copy
RETURN

```

ENDIF ; thisEvent = UP

; Most of the action in this event handler occurs here while we are waiting for an UP
; event to occur. As long as we don't get it, keep erasing the old zoom box and drawing a
; new one.


```

; Erase the old zoom box.
WSet, info.widl
Device, Copy = [0, 0, 512, 512, 0, 0, info.boxwid]

; Update the dynamic corner of the zoom box to the current cursor location.
newx = event.x
newy = event.y

; Load a color in color index 1 to draw the zoom box with.
TVLct, 255B, 0B, 0B, 1

; Draw the zoom box.
PlotS, [info.mousex, info.mousex, newx, newx, info.mousex], $
      [info.mousey, newy, newy, info.mousey, info.mousey], /Device, Color=1

widget_control, info.drawlid, set_uvalue=recon, /no_copy

; Put the info structure back into its storage location.
Widget_Control, event.top, Set_UValue=info, /No_Copy

end ; of zoommove_event
; *****

PRO dhdouble_mask_Event, event

widget_control,event.top,get_uvalue=info
print, info
widget_control,event.id,get_uvalue=value
widget_control,info.holoid,get_uvalue=holo
help, /memory

CASE value OF
    'DIM' : begin ;obtain image X & Y size
        widget_control,event.top,tlb_get_offset=offsets
        imdim=getdim(offsets(0)+50,offsets(1)+75, info.m, info.n)
        if imdim.xsize ne 0 then begin
            info.m=imdim.xsize
            info.n=imdim.ysize
            info.mc=imdim.xsize
            info.nc=imdim.ysize
            info.xscale=double(info.mc/512.)
            info.yscale=double(info.nc/512.)
            print,'info.yscale',info.yscale,size(info.yscale)
            widget_control,event.top,set_uvalue=info
        endif
    endif

```



```

end
'ORIENT' : begin ;obtain orientation of plane wave
    widget_control,event.top,tlb_get_offset=offsets
    orientdim=orient(offsets(0)+50,offsets(1)+75,info.theta,info.phi)
    info.theta=double(orientdim.theta)
    info.phi=double(orientdim.phi)
    widget_control,event.top,set_uvalue=info
end
'D' : begin ;obtain distance between holo and object
    widget_control,event.top,tlb_get_offset=offsets
    info.d=getd(offsets(0)+50,offsets(1)+75,info.d)
    info.d=double(info.d)
    widget_control,event.top,set_uvalue=info
end
'MISC' : begin
    widget_control,event.top,tlb_get_offset=offsets
    misc=getmisc(offsets(0)+50,offsets(1)+75,info.dx,info.dy $
        ,info.lambda,info.mag)
    info.dx=double(misc.dx)
    info.dy=double(misc.dy)
    info.lambda=double(misc.lambda)
    info.mag=double(misc.mag)
    widget_control,event.top,set_uvalue=info
end
'LOAD' :begin ;read interference image
    widget_control,event.top,tlb_get_offset=offsets
    pick=dialog_pickfile(/read, path=info.impath, get_path=impath)
    info.impath=impath
    info.pick=pick

    ;determine the image type
    pick=strlowercase(pick)
    type=strmid(pick,strlen(pick)-3,3)
    if type eq 'byt' then imtype=0
    if type eq 'tif' then imtype=1
    if type eq 'byt' or type eq 'tif' then begin
        info.imtype=imtype
        holo=read_dh(info.pick,info.imtype, info.m, info.n, info.wid1)
        widget_control, event.top, set_uvalue=info
        widget_control, info.holoid, set_uvalue=holo, /no_copy
    endif
end
end
'RELOAD' :begin

```



```
; the read program reads the current file and displays it, it returns holo
holo=read_dh(info.pick,info.imtype, info.m, info.n, info.wid1)
```

```
info.mc=info.m
info.nc=info.n
widget_control,event.top,set_uvalue=info
widget_control, info.holoid, set_uvalue=holo, /no_copy
end
```

```
'CUT' :begin
```

```
widget_control,event.top,tlb_get_offset=offsets
```

```
; send mc & nc so that present values can be displayed
cutdata=imcutarr(offsets(0)+50, offsets(1)+75, info.mc, info.nc, $
info.pick,info.imtype, info.m, info.n, info.wid1)
```

```
if cutdata.accept eq 0 then begin ;use current dimensions
```

```
info.mc=info.mc
```

```
info.nc=info.nc
```

```
endif
```

```
if cutdata.accept eq 1 then begin; if not 1, then rest is skipped
```

```
info.mc=cutdata.newxsize
```

```
info.nc=cutdata.newysize
```

```
info.xscale=double(info.mc/512.)
```

```
info.yscale=double(info.nc/512.)
```

```
if cutdata.newcenter eq 1 then begin
```

```
cursor, subx, suby, /device, /down
```

```
info.suby=suby
```

```
info.subx=subx
```

```
endif
```

```
holo=extrac(holo,info.subx*info.m/512-info.mc/2, $
```

```
info.suby*info.n/512-info.nc/2,info.mc,info.nc)
```

```
widget_control,info.holoid,set_uvalue=holo
```

```
wset, info.wid1
```

```
erase
```

```
tvsc1,congrid(holo,512,512)
```

```
endif
```

```
widget_control,event.top,set_uvalue=info
```

```
end
```

```
'RELOADCUT' :begin
```

```
;read byte
```

```
if info.imtype eq 0 then begin
```

```
openr,1,info.pick
```

```
holo=bytarr(info.m,info.n)
```

```
readu,1,holo
```



```

        close,1
    endif

    ;read tiff
    if info.imtype eq 1 then begin
        holo=tiff_read(info.pick)
    endif

    ; extract the subimage from the original image
    holo=extrac(holo,info.subx*info.m/512-info.mc/2, $
    info.suby*info.n/512-info.nc/2,info.mc,info.nc)

    wset, info.wid1
    erase
    tvscl,congrid(holo,512,512)
    widget_control, info.holoid, set_uvalue=holo, /no_copy
    end

'SAVE' :begin
    widget_control,info.draw1id,get_uvalue=recon
    if info.disptype eq 2 then begin
        recon=alog(double(recon)+1d-6)
    endif
    recon=bytsc1(recon)
    pick=dialog_pickfile(/write, path=info.impath, get_path=impath)
    info.impath=impath
    tiff_write,pick,recon
    widget_control,event.top,set_uvalue=info
    end

'MASK' :begin
    widget_control, info.draw1id, draw_Button_Events=1, $
        event_pro='mask_event'
    wset,info.boxwid
    device, copy = [0, 0, 512, 512, 0, 0, info.wid1]
    widget_control, info.draw1id, set_uvalue=recon, /no_copy
    widget_Control, event.top, set_uvalue=info
    end

'LINELOG' :begin
    widget_control,event.top,tlb_get_offset=offsets
    widget_control,info.draw1id,get_uvalue=recon
    disptype=linealog(offsets(0)+50, offsets(1)+75)
    if disptype eq 1 then begin
        wset,info.wid1
        erase
        tvscl,congrid(recon,512,512)
        wset,info.wid2
    end
end

```



```

        erase
        tvscl,recon(info.zoomxlow:info.zoomxhigh, $
                    info.zoomylow:info.zoomyhigh)
    endif
    if disptype eq 2 then begin
        wset,info.wid1
        erase
        tvscl,congrid(alog(double(recon)+1d-6),512,512)
        wset,info.wid2
        erase
        tvscl,alog(double(recon(info.zoomxlow:info.zoomxhigh, $
                    info.zoomylow:info.zoomyhigh))+1d-6)
    endif
    info.disptype=disptype
    widget_control,event.top,set_uvalue=info
end
'RECON':begin

;calculate planewave and gaussian
k=2.*!dpi/info.lambda
kx=k*cos(info.theta*!dpi/180d)*sin(info.phi*!dpi/180d)
ky=k*sin(info.theta*!dpi/180d)*sin(info.phi*!dpi/180d)
vectx=(dindgen(info.mc)-round(info.mc/2))*info.dx/info.mag
arx=transpose(make_array(info.nc,1,/double,value=1.0))
vecty=(transpose(dindgen(info.nc)-round(info.nc/2)))*info.dy/info.mag
ary=make_array(info.mc,1,/double,value=1.0)
planewave=kx*(vectx#arx)+ky*(ary#vecty)
planewave=exp(dcomplex(0,temporary(planewave)))
gauss=-((vectx#arx)^2+(ary#vecty)^2)*k/2.d/info.d
gauss=exp(dcomplex(0,temporary(gauss)))

recon=double(holo)*gauss*planewave
gauss=0
planewave=0

; Positive # indicates an inverse Fourier transform
recon=fft(temporary(recon),-1,double=1,/overwrite)

recon=abs(temporary(recon))^2
recon=shift(temporary(recon),info.mc/2,info.nc/2)
beep
if info.disptype eq 1 then begin
    wset,info.wid1
    tvscl,congrid(recon,512,512)
    wset,info.wid2

```



```

        tvscl,congrid(recon(info.zoomxlow:info.zoomxhigh, $
            info.zoomylow:info.zoomyhigh),150,150)
    endif
    if info.disptype eq 2 then begin
        wset,info.wid1
        tvscl,congrid(alog(double(recon))+1d-6,512,512)
        wset,info.wid2
        tvscl,congrid(alog(double(recon(info.zoomxlow:info.zoomxhigh, $
            info.zoomylow:info.zoomyhigh))+1d-6),150,150)
    endif
    widget_control,event.top,set_uvalue=info
    widget_control,info.holoid,set_uvalue=holo, /no_copy
    widget_control,info.drawlid,set_uvalue=recon, /no_copy
end
'ZOOM' :begin
    widget_control, info.drawlid, draw_Button_Events=1, $
        event_pro='zoom_event'
    wset,info.boxwid
    device, copy = [0, 0, 512, 512, 0, 0, info.wid1]
    widget_control, info.drawlid, set_uvalue=recon, /no_copy
    widget_Control, event.top, set_uvalue=info
end
'EXIT' : Widget_Control, event.top, /Destroy
ENDCASE

```

END ; of dhdouble_mask_Event

; *****

PRO dhdouble

; images are transferred from bottom to top, i.e. the row with a 0 subscript is drawn on
; the bottom
!order=0

```

device, retain=2
base = Widget_Base(Title='Digital Holography (double precision)', Column=2)
drawbase=widget_base(base,column=2)
btnbase=widget_base(base,column=1)
btn1=widget_button(btnbase,value='Input Image Dimensions',uvalue='DIM')
btn2=widget_button(btnbase,value='Input Plane Orientation',uvalue='ORIENT')
btn3=widget_button(btnbase,value='Input D',uvalue='D')
btn13=widget_button(btnbase,value='Input dx, dy, lambda, mag',uvalue='MISC')
btn4=widget_button(btnbase,value='Load Interference Image',uvalue='LOAD')
btn5=widget_button(btnbase,value='Re-Load Interference Image',uvalue='RELOAD')
btn6=widget_button(btnbase,value='Extract Partial Image',uvalue='CUT')

```



```

btn11=widget_button(btnbase,value='Re-Load Extracted Image',uvalue='RELOADCUT')
btn7=widget_button(btnbase,value='Reconstruct Hologram',uvalue='RECON')
btn12=widget_button(btnbase,value='Linear/Log Display',uvalue='LINELOG')
btn14=widget_button(btnbase,value='Zoom',uvalue='ZOOM')
btn10=widget_button(btnbase,value='Mask Image Portion',uvalue='MASK')
btn8=widget_button(btnbase,value='Save Reconstructed Image (TIF)',uvalue='SAVE')
btn9=widget_button(btnbase,value='Exit',uvalue='EXIT')
draw1=widget_draw(drawbase,xsize=512,ysize=512)
draw2=widget_draw(btnbase,xsize=150,ysize=150)
Widget_Control, base, /Realize
widget_control,draw1,get_value=wid1
widget_control,draw2,get_value=wid2
Window, /Free, XSize=512, YSize=512, /Pixmap
boxwid = !D.Window
xscale=1024d/512d
yscale=1024d/512d
info={m:1024, n:1024, mc:1024, nc:1024, theta:0d,phi:0d, d:0.218d, $
      wid1:wid1, wid2:wid2, holoid:drawbase, draw1id:draw1, imtype:0, $
      disptype:2, subx:128l, suby:128l, pick:"", boxwid:boxwid, mousex:0, $
      mousey:0, r_old:0, g_old:0, b_old:0, maskxlow:0, maskxhigh:0, $
      maskylow:0, maskyhigh:0, xscale:xscale, yscale:yscale, impath:'e:\', $
      mag:7.4664d, lambda:632.8d-9, dx:6.8d-6, dy:6.8d-6, zoomxlow:0, $
      zoomxhigh:127, zoomylow:0, zoomyhigh:127}
holo=0b
recon=0b
loadct,0
wset,wid1
erase
widget_control,base,set_uvalue=info
widget_control,drawbase,set_uvalue=holo
widget_control,draw1,set_uvalue=recon
XManager, 'dhdouble_mask', base, Event='dhdouble_mask_Event'
END ; of PRO dhdouble

```

```

, *****
,
, *****
,

```

The source code for *getd.pro* is the following:

```

, *****
,
, *****
,

```

Pro getd_Event, event


```

name = Tag_Names(event, /Structure_Name)
IF name EQ 'WIDGET_BUTTON' THEN BEGIN
    Widget_Control, event.top, Get_UValue=info, /No_Copy
    Widget_Control, event.id, Get_Value=test

    CASE test OF
        'CANCEL'      : Begin
            data = info.currentd
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            End
        'ACCEPT'      : Begin
            Widget_Control, info.d_id, Get_Value=d
            data = d
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            End
    ENDCASE

ENDIF
END ; of getd_Event
; *****

```

Function getd, xoff, yoff, currentd

```

base = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, $
Title='Enter D Value')
filebase = Widget_Base(base, Column=1)
filelabel = Widget_Label(filebase, Value='Input D')
d_id = CW_Field(base, Title='D: ', Value=currentd, /float, /Return_Events)
btnbase = Widget_Base(base, Row=1, /Frame)
cancel = Widget_Button(btnbase, Value='CANCEL')
done = Widget_Button(btnbase, Value='ACCEPT')
Widget_Control, base, /Realize
handle = handle_create()
info = {d_id:d_id, handle:handle, currentd:currentd}
Widget_Control, base, Set_UValue=info, /No_Copy
XManager, 'getd', base, /Modal

```

;Must use handle because all widgets are destroyed after Xmanager.
Handle_Value, handle, data, /No_Copy

```

Handle_Free, handle
Return, data
End ; Function getd

```



```

, *****
, *****
,

```

The source code for *getdim.pro* is the following:

```

, *****
, *****
,

```

Pro getdim_Event, event

```

name = Tag_Names(event, /Structure_Name)
IF name EQ 'WIDGET_BUTTON' THEN BEGIN
    Widget_Control, event.top, Get_UValue=info, /No_Copy
    Widget_Control, event.id, Get_Value=test

    CASE test OF
        'CANCEL'      : Begin
            data={xsize:0}
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            End
        'ACCEPT'      : Begin
            Widget_Control, info.xsize_id, Get_Value=xsize
            Widget_Control, info.ysize_id, Get_Value=ysize
            data = {xsize:xsize, ysize:ysize}
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            End
    ENDCASE
ENDIF
END ; of getdim_Event
, *****

```

Function getdim, xoff, yoff, currentx, currenty

```

base = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, Title='Enter File Sizes')
filebase = Widget_Base(base, Column=1)
filelabel = Widget_Label(filebase, Value='Input Image Dimensions')
xsize_id = CW_Field(base, Title='X Size: ', Value=currentx, /Integer, /Return_Events)
ysize_id = CW_Field(base, Title='Y Size: ', Value=currenty, /Integer, /Return_Events)
btnbase = Widget_Base(base, Row=1, /Frame)
cancel = Widget_Button(btnbase, Value='CANCEL')

```



```

done = Widget_Button(btnbase, Value='ACCEPT')
Widget_Control, base, /Realize
handle = handle_create()
info = {xsize_id:xsize_id, ysize_id:ysize_id, handle:handle, currentx:currentx, $
        currenty:currenty}
Widget_Control, base, Set_UValue=info, /No_Copy
XManager, 'getdim', base, /Modal
Handle_Value, handle, data, /No_Copy
Handle_Free, handle
Return, data
End ; of Function getdim

```

```

, *****
, *****
, *****

```

The source code for *getmisc.pro* is the following:

```

, *****
, *****
, *****

```

Pro getmisc_Event, event

```

name = Tag_Names(event, /Structure_Name)
IF name EQ 'WIDGET_BUTTON' THEN BEGIN
    Widget_Control, event.top, Get_UValue=info, /No_Copy
    Widget_Control, event.id, Get_Value=test

    CASE test OF
        'CANCEL'      : Begin
            data={dx:currentdx,dy:currentdy,lambda:currentlambda, $
                  mag:currentmag}
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            End
        'ACCEPT'      : Begin
            Widget_Control, info.dx_id, Get_Value=dx
            Widget_Control, info.dy_id, Get_Value=dy
            Widget_Control, info.lambda_id, Get_Value=lambda
            Widget_Control, info.mag_id, Get_Value=mag
            data = {dx:dx,dy:dy,lambda:lambda,mag:mag}
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            End
    ENDCASE

```



```

ENDIF
END ; of getmisc_Event
; *****

Function getmisc, xoff, yoff, currentdx, currentdy, currentlambda, currentmag
base = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, $
    Title='Miscellaneous Parameters')
filebase = Widget_Base(base, Column=1)
filelabel = Widget_Label(filebase, Value='Input Parameters')
dx_id = CW_Field(base, Title='dx: ', Value=currentdx, /float, /Return_Events)
dy_id = CW_Field(base, Title='dy: ', Value=currentdy, /float, /Return_Events)
lambda_id = CW_Field(base, Title='Lambda: ', Value=currentlambda, $
    /float, /Return_Events)
mag_id = CW_Field(base, Title='Mag: ', Value=currentmag, /float, /Return_Events)
btnbase = Widget_Base(base, Row=1, /Frame)
cancel = Widget_Button(btnbase, Value='CANCEL')
done = Widget_Button(btnbase, Value='ACCEPT')
Widget_Control, base, /Realize
handle = handle_create()
info = {dx_id:dx_id, dy_id:dy_id, lambda_id:lambda_id, mag_id:mag_id, handle:handle}
Widget_Control, base, Set_UValue=info, /No_Copy
XManager, 'getmisc', base, /Modal
Handle_Value, handle, data, /No_Copy
Handle_Free, handle
Return, data
End ; of Function getmisc

; *****
; *****

```

The source code for *imcutarr.pro* is the following:

```

; *****
; *****

```

Pro imcutarr_Event, event

```

name = Tag_Names(event, /Structure_Name)
IF name EQ 'WIDGET_BUTTON' THEN BEGIN
    Widget_Control, event.top, Get_UValue=info, /No_Copy
    Widget_Control, event.id, Get_Value=test

    CASE test OF

```



```

        'ACCEPT'      : Begin
            Widget_Control, info.newxsize_id, Get_Value=newxsize
            Widget_Control, info.newysize_id, Get_Value=newysize
            data = {newxsize:newxsize,newysize:newysize, $
                    accept:1,newcenter:1}
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            wshow, info.wid1
            widget_Control, info.base2_id, /realize
            Widget_Control, info.base2_id, Set_UValue=info
            XManager, 'redisplay', info.base2_id, /Modal
            End
        'CANCEL'      : Begin
            data = {accept:0}
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            wshow, info.wid1
            End
    ENDCASE

ENDIF
END ; of imcutarr_Event
; *****

Pro redisplay_Event, event

name = Tag_Names(event, /Structure_Name)
IF name EQ 'WIDGET_BUTTON' THEN BEGIN
    Widget_Control, event.top, Get_UValue=info, /No_Copy
    Widget_Control, event.id, Get_Value=test

    CASE test OF
        'NO'      : Begin
            Widget_Control, event.top, /Destroy
            wshow, info.wid1
            widget_Control, info.base3_id, /realize
            Widget_Control, info.base3_id, Set_UValue=info
            XManager, 'mouseinfo', info.base3_id, /Modal
            end
        'YES'      : Begin
            holo=read_dh(info.file, info.imtype, info.m, info.n, info.wid1)
            Widget_Control, event.top, /Destroy
            wshow, info.wid1
            widget_Control, info.base3_id, /realize
            Widget_Control, info.base3_id, Set_UValue=info
    
```



```

                                XManager, 'mouseinfo', info.base3_id, /Modal
                                end
                                ENDCASE

endif
End ; redisplay_Event
; *****

Pro mouseinfo_Event, event

name = Tag_Names(event, /Structure_Name)
IF name EQ 'WIDGET_BUTTON' THEN BEGIN
    Widget_Control, event.top, Get_UValue=info, /No_Copy
    Handle_Value, info.handle, data, /No_Copy
    Widget_Control, event.id, Get_Value=test

    CASE test OF
        'CANCEL'      : Begin
                        data.newcenter=0
                        Handle_Value, info.handle, data, /Set, /No_Copy
                        Widget_Control, event.top, /Destroy
                        wshow, info.wid1
                        end
        'OK'          : Begin
                        data.newcenter=1
                        Handle_Value, info.handle, data, /Set, /No_Copy
                        Widget_Control, event.top, /Destroy
                        wshow, info.wid1
                        end
    ENDCASE

endif
End ; mouseinfo_Event
; *****

Function imcutarr, xoff, yoff, currentxsize, currentysize, file, imtype, m, n, wid1

base = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, $
    Title='Cut Interference Image')
filebase = Widget_Base(base, Column=1)
filelabel = Widget_Label(filebase, Value='Specify New Image Dimensions')
base3 = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, $
    Title='Select Image Center')
filebase3 = Widget_Base(base3, Column=1)
filelabel3 = Widget_Label(base3, Value='Click Mouse At Desired Center')

```



```

ok = Widget_Button(base3, Value='OK')
can=Widget_Button(base3, Value='CANCEL')
base2=Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, Title='Redisplay')
filelabel2=Widget_Label(base2, Value='Redisplay Original Image?')
yes=Widget_Button(base2, Value='YES')
no=Widget_Button(base2, Value='NO')
newxsize_id = CW_Field(base, Title='New X Size: ', Value=currentxsize, $
    /long, /Return_Events)
newysize_id = CW_Field(base, Title='New Y Size: ', Value=currentysize, $
    /long, /Return_Events)
btnbase = Widget_Base(base, Row=1, /Frame)
cancel = Widget_Button(btnbase, Value='CANCEL')
done = Widget_Button(btnbase, Value='ACCEPT')
Widget_Control, base, /Realize
handle = handle_create()
info = {newxsize_id:newxsize_id,newysize_id:newysize_id,handle:handle, $
    base2_id:base2, base3_id:base3,currentxsize:currentxsize, $
    currentysize:currentysize, file:file, imtype:imtype, m:m, n:n, wid1:wid1}
Widget_Control, base, Set_UValue=info, /No_Copy
XManager, 'imcutarr', base, /Modal
Handle_Value, handle, data, /No_Copy
Handle_Free, handle
Return, data
End ; of Function imcutarr

```

```

; *****
; *****
;

```

The source code for *linelog.pro* is the following:

```

; *****
; *****
;

```

Pro linelog_Event, event

```

name = Tag_Names(event, /Structure_Name)
IF name EQ 'WIDGET_BUTTON' THEN BEGIN
    Widget_Control, event.top, Get_UValue=handle
    Widget_Control, event.id, Get_Value=test

    CASE test OF
        'LINEAR' :Begin
            Handle_Value, handle, 1, /Set
            Widget_Control, event.top, /Destroy

```



```

        End
    'LOG' :Begin
        Handle_Value, handle, 2, /Set
        Widget_Control, event.top, /Destroy
    End
    'CANCEL' :Begin
        Handle_Value, handle, 3, /Set
        Widget_Control, event.top, /Destroy
    End
ENDCASE

ENDIF
END ; of Pro linelog
; *****

function linelog, xoff, yoff

base = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, $
    Title='Linear/Log Display')
filebase = Widget_Base(base, Column=1)
filelabel = Widget_Label(filebase, Value='Select Display Method')
linear = Widget_Button(base, Value='LINEAR')
log = Widget_Button(base, Value='LOG')
cancel = Widget_Button(base, Value='CANCEL')
Widget_Control, base, /Realize
handle = handle_create()
Widget_Control, base, Set_UValue=handle
XManager, 'linelog', base, /Modal
Handle_Value, handle, data, /No_Copy
Handle_Free, handle
Return, data
End ; function linelog

; *****
; *****
; *****

```

The source code for *orient.pro* is the following:

```

; *****
; *****
; *****

Pro orient_Event, event

name = Tag_Names(event, /Structure_Name)

```



```

IF name EQ 'WIDGET_BUTTON' THEN BEGIN
    Widget_Control, event.top, Get_UValue=info, /No_Copy
    Widget_Control, event.id, Get_Value=test

    CASE test OF
        'CANCEL'      : Begin
            data = {theta:currenttheta,phi:currentphi}
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            End
        'ACCEPT'      : Begin
            Widget_Control, info.theta_id, Get_Value=theta
            Widget_Control, info.phi_id, Get_Value=phi
            data = {theta:theta,phi:phi}
            Handle_Value, info.handle, data, /Set, /No_Copy
            Widget_Control, event.top, /Destroy
            End
    ENDCASE

ENDIF
END ; of orient_Event
; *****

Function orient, xoff, yoff, currenttheta, currentphi

base = Widget_Base(Column=1, XOffset=xoff, YOffset=yoff, $
    Title='Plane Wave Orientation')
filebase = Widget_Base(base, Column=1)
filelabel = Widget_Label(filebase, Value='Input Plane Orientation (degrees)')
theta_id = CW_Field(base, Title='Theta: ', Value=currenttheta, /float, /Return_Events)
phi_id = CW_Field(base, Title='Phi: ', Value=currentphi, /float, /Return_Events)
btnbase = Widget_Base(base, Row=1, /Frame)
cancel = Widget_Button(btnbase, Value='CANCEL')
done = Widget_Button(btnbase, Value='ACCEPT')
Widget_Control, base, /Realize
handle = handle_create()
info = {theta_id:theta_id, phi_id:phi_id, handle:handle}
Widget_Control, base, Set_UValue=info, /No_Copy
XManager, 'orient', base, /Modal
Handle_Value, handle, data, /No_Copy
Handle_Free, handle
Return, data
End ; of Function orient

; *****

```



```
, *****
;
```

The source code for *read_dh.pro* is the following:

```
, *****
, *****
;
```

Function read_dh, file, imtype, m, n, wid1

```
;read byte
if imtype eq 0 then begin
    openr,1,file
    holo=bytarr(m,n)
    readu,1,holo
    close,1
endif

;read tiff
if imtype eq 1 then begin
    holo=tiff_read(file)
    print,size(holo)
endif
holo=reverse(holo,2); reverses the rows
wset, wid1
erase
tvscl, congrid(holo,512,512)
return, holo
end ; of Function read_dh
```

```
, *****
, *****
;
```


APPENDIX D

Relative Error Propagation

A key element of interpreting the experimental data is error analysis. This section will be devoted to a short summary of error analysis, and in particular, relative error propagation. First of all, relative error, $\varepsilon_{\tilde{x}_i}$, is defined as [Stoer, Bulirsch, 1993, p. 12]

$$\varepsilon_{\tilde{x}_i} = \frac{\tilde{x}_i - x_i}{x_i}, \quad (\text{D.1})$$

where \tilde{x}_i is the estimate or measured value of x_i , and x_i is the true value. Once the relative errors of all variables are known, the total relative error must be determined. This total relative error depends on the mathematical operations performed in reaching the final result.

The equation which applies in determining relative error propagation resulting from a mathematical operation is given by the first order approximation [Stoer, Bulirsch, 1993, p. 13]

$$\varepsilon_{y_i} = \sum_{j=1}^n \frac{x_j}{\phi_i(x)} \frac{\partial \phi_i(x)}{\partial x_j} \varepsilon_{x_j}. \quad (\text{D.2})$$

For example, if

$$\phi(x, y) = x \cdot y, \quad (\text{D.3})$$

where the variable x has a relative error of ε_x and the variable y has a relative error of ε_y , then the relative of $\phi(x, y)$ is given by [Stoer, Bulirsch, 1993, p. 14]

$$\begin{aligned} \varepsilon_{\phi(x,y)} &= \frac{x}{xy} \frac{\partial(xy)}{\partial x} \varepsilon_x + \frac{y}{xy} \frac{\partial(xy)}{\partial y} \varepsilon_y \\ &= \frac{1}{y} y \varepsilon_x + \frac{1}{x} x \varepsilon_y \\ &= \varepsilon_x + \varepsilon_y. \end{aligned} \quad (\text{D.4})$$

In a similar fashion, the following relative error propagation formulas for common mathematical operations can be computed from application of Eq. (D.2) [Stoer, Bulirsch, 1993, p. 14]:

$$\phi(x, y) = \frac{x}{y} \Rightarrow \varepsilon_{\phi(x,y)} = \varepsilon_x - \varepsilon_y \quad (\text{D.5})$$

$$\phi(x, y) = x + y \Rightarrow \varepsilon_{\phi(x,y)} = \frac{x}{x+y} \varepsilon_x + \frac{y}{x+y} \varepsilon_y \quad (\text{D.6})$$

$$\phi(x, y) = x - y \Rightarrow \varepsilon_{\phi(x,y)} = \frac{x}{x-y} \varepsilon_x - \frac{y}{x-y} \varepsilon_y. \quad (\text{D.7})$$

Care must be taken in applying Eq. (D.5). At first, it may seem that the relative errors are simply subtracted. However, this equation actually is

$$\phi(x, y) = \frac{x}{y} \Rightarrow \varepsilon_{\phi(x, y)} = (\pm|\varepsilon_x|) - (\pm|\varepsilon_y|) = \pm(|\varepsilon_x| + |\varepsilon_y|). \quad (\text{D.8})$$

One final point must be made regarding relative errors. The point is that in order to calculate the relative error as defined in Eq. (D.1), the true value of the variables must be known. Obviously, if the true values are known, then there are no errors to be concerned about. Therefore, a method of estimating the relative error must be used. A reasonable approximation to use is to use the estimate or measured value of the variables in the denominator of Eq. (D.1). In addition, the numerator of Eq. (D.1), known as the absolute error, is found by applying a reasonable estimate of how close the measured value is to the true value. A good example is found in the case of measuring a length with a ruler. Generally, it is reasonable to assume that a measurement can be made to within the length of the smallest scale on the ruler. For example, if the smallest scale is 1mm, then a good estimate is that the true value should be within $\pm 0.5\text{mm}$.

LIST OF REFERENCES

- Born, M., Wolf, E., Principles of Optics, Sixth (Corrected) Edition, Cambridge University Press, 1980.
- Clark, P. P., Howard, J. W., and Freniere, E. R., "Asymptotic Approximation to the Encircled Energy Function for Arbitrary Aperture Shapes", Applied Optics, Volume 23, Issue 2, January 15, 1984, p. 353-357.
- Goodman, J. W., Introduction to Fourier Optics, Second Edition, The McGraw-Hill Companies, Inc., 1996.
- Hecht, E., Optics, Third Edition, Addison Wesley Longman, Inc., 1998.
- Kozma, A., "Photographic Recording of Spatially Modulated Coherent Light", Journal of the Optical Society of America, 56, (1966), p. 428-432.
- Kreis, T., Holographic Interferometry: Principles and Methods, Akademie Verlag GmbH, Berlin, 1996.
- Longhurst, R. S., Geometrical and Physical Optics, Second Edition, New York, Wiley, 1967.
- Matthews, P. A., Cullen, A. L., "A Study of the Field Distribution at an Axial Focus of a Square Microwave Lens", Proceedings of The Institution of Electrical Engineers Part. C, 103, Monograph No. 186 R, July 1956, p. 449-453.
- Mouroulis, P., Zhang, H., "Visual Instrument Image Quality Metrics and the Effects of Coma and Astigmatism", Journal of the Optical Society of America A, Vol. 9, No.1, January 1992, p. 34-42.
- Reynolds, G. O., Parrent, G. B., and Thompson, B. J., The New Physical Optics Notebook: Tutorials in Fourier Optics, Bellingham, Wash., USA: Optical Engineering Press, 1989.
- Rivolta, C., "Airy Disk Diffraction Pattern: Comparison of Some Values of F/No. and Obscuration Ratio", Applied Optics, Volume 25, Issue 14, July 15, 1986, p. 2404-2408.
- Slepian, D., "Analytic Solution of Two Apodization Problems", Journal of the Optical Society of America, Vol. 55, No. 9, September 1965, p. 1110-1115.
- Stoer, J., Bulirsch, R., Introduction to Numerical Analysis, Second Edition, Springer-Verlag New York, Inc., 1993.

Strum, R. D., Kirk, D. E., First Principles of Discrete Systems and Digital Signal Processing, Addison-Wesley Publishing Company, Inc., 1989.

Vikram, C., Particle Field Holography, Cambridge University Press, 1992.

Yaroslavskii, L. P., Merzlyakov, N. S., Methods of Digital Holography, Consultants Bureau, 1980.

Ziemer, R. E., Tranter, W. H., and Fannin, D. R., Signals and Systems: Continuous and Discrete, Macmillan Publishing Company, Inc., 1983.

DISTRIBUTION LIST
AFRL-MN-EG-TR-2000-7045

Defense Technical Information Center
8725 John J. Kingman Road, Ste 0944
Ft Belvoir, VA 22060-6218

1

Air University Library
600 Chennault Circle
Bldg 1405
Maxwell AFB, AL 36112-6424

1

EGLIN AFB OFFICES:

AFRL/MN/CA-N

1

AFRL/MNOC-1 (Technical Library)

1

AFRL/MNA

1

AFRL/MNG

1

AFRL/MNM

1

AFRL/MNAL

10